VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Návrh a implementace univerzálního datového rozhraní pro prohlížeč reálných a historických trendů v prostředí .NET

Design and Implementation of Universal Data Interface for Actual and Historical Trending in .NET

2013 Martin Mikolajek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání diplomové práce

Student: **Bc. Martin Mikolajek**

Studijní program: N2649 Elektrotechnika

Studijní obor: 2601T004 Měřicí a řídicí technika

Téma: Návrh a implementace univerzálního datového rozhraní pro prohlížeč reálných a historických trendů v prostředí .NET
Design and Implementation of Universal Data Interface for Actual and Historical Trending in .NET

Zásady pro vypracování:

1. Rozbor současných řešení zobrazování dat z řídicích systémů ve formě trendů.
2. Přehled a analýza dostupných produktů pro přístup ke zdrojovým datům.
3. Návrh univerzálního datového rozhraní pro prohlížeč reálných a historických trendů.
4. Implementace v prostředí .NET.
5. Testování funkčnosti systému na vzorcích dat.
6. Zhodnocení výsledků.
7. Rozšířený abstrakt v anglickém jazyce v rozsahu 3 strany sumarizující řešení. Rozšířený abstrakt bude přiložen jako jedna z příloh.

Seznam doporučené odborné literatury:

[1] NAGEL, Christian. *C# 2008: programujeme profesionálně.* Vyd. 1. Brno: Computer Press, 2009. 772 s. ISBN 978-80-251-2401-7.
[2] BERGER, Hans. *Automating with SIMATIC: controllers, software, programming, data communication, operator control and process monitoring.* 4th rev. and enlarged ed. Erlangen: Publicis Pub., 2009. 236 s. ISBN 978-389-5783-333-3.
3. Firemní technická dokumentace pro řídicí systémy firmy Siemens.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Jiří Koziorek, Ph.D.**

Datum zadání: 18.11.2011
Datum odevzdání: 07.05.2013

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

„Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal."

.....

Martin Mikolajek

Datum odevzdání diplomové práce 7.5.2013

Poděkování

Děkuji všem, kteří mi pomohli při tvorbě této práce a také mé rodině za podporu ve studiu.

Abstrakt

Diplomová práce se zabývá návrhem a vytvořením univerzálního datového rozhraní pro prohlížeč historických a reálných trendů v průmyslové automatizaci. Datové rozhraní je implementováno v programovacím jazyku C# formou dynamické knihovny (DLL). Téma diplomové práce vzniklo z požadavků firmy Ingeteam a.s., která specifikovala zadání, navrhnout univerzální datové rozhraní pro trendový prohlížeč za pomoci využití platformy .NET. Vytvořené univerzální datové rozhraní umožňuje základní konfiguraci jednotlivých datových zdrojů jako například, výběr počtu signálů nebo vzorkovací periodu. Při vytváření datového rozhraní byl kladen důraz na univerzálnost, pro případnou možnost rozšíření přístupů k dalším datovým zdrojům. Z tohoto důvodu univerzálnosti obsahuje datové rozhraní základní předpis komunikačních funkcí neboli interface, který slouží trendovému prohlížeči pro vyčítání dat z jednotlivých datových zdrojů.

Klíčová slova

datové rozhraní, .NET Framework, dynamická knihovna, PLC, trendový prohlížeč, signál

Abstract

This diploma thesis is focused on design and implementation of universal data interface for historical and real trends viewer analyzer in industry automation using. This data interface is implemented in C# programming language as (DLL) dynamic link library. Theme this master work came from Ingeteam a.s. company request. The company‘s request is design and create universal data interface for trend analyzer by using .NET technology. This implemented data interface allows basic configuration for all data sources. For example - selectable count signals or signal sample step. During creating data interface the emphasis was on universality. That is mean that, data interface can be extended by reading from more data inputs. The universality is the reason why the trend analyzer is communicating with data interface with clearly defined functions.

Keyword

data interface, .NET framework, dynamic library, PLC, trend viewer, signal

Seznam použitých symbolů a zkratek

.NET	Softwarová technologie, platforma Microsoftu
C#	Objektový programovací jazyk pro platformu .NET
CD	Collision Detection
CLR	Common Language Runtime
CSMA	Carrier Sense Multiple Access
CSV	Comma-separated values datový formát pro ukládání tabulkových dat
DBase	Database management system
DDE	Dynamic Data Exchange
DLL	Dynamic Link Library
GUI	Graphical user interface
HTML	HyperText Markup Language
IDE	Integrated development environment
IEC	International Electrotechnical Commission
IP	Internetový protokol
ISO/OSI	International Standards Organization / Open Systen Interconnection
MPI	Multipoint Interface - komunikační rozhraní pro průmyslová zařízení
MySQL	Multiplatformní databáze SQL
NetDDE	Network Dynamic Data Exchange
NetPro	Konfigurátor sítě
ODBC	Open Database Connectivity
OLE DB	Object Linking and Embedding Database
OOP	Object-oriented programming
OPC	Object Linking and Embedding for Process Control
PC	Persobnal computer
PLC	Programmable Logic Controller
PPI	Point-to-Point Interface - komunikační rozhraní pro průmyslová zařízení
RS485	Standard sériové komunikace
SCADA	Supervisory control and data acquisition
SQL	Structured Query Language
TCP/IP	Sadu protokolů pro komunikaci v počítačové síti
TXT	Formát textového souboru
XLS	Přípona souborů vytvořených v aplikaci Microsoft Excel
XML	Extensible Markup Language

Obsah

1 Úvod

Tato diplomová práce se zabývá návrhem a implementací datového rozhraní pro trendový prohlížeč reálných a historických trendů s využitím v průmyslové automatizaci. Podnět pro tvorbu datového rozhraní, které je tématem této diplomové práce přichází z firmy Ingeteam a.s, která se zabývá automatizací v průmyslu. Při uvádění PLC systémů pro průmyslové zařízení jsou pro firmu důležité nástroje, které slouží pro přehledné sledování procesních hodnot v řízených technologických procesech. V dnešní době se pro průmyslovou automatizaci využívají profesionální datové prohlížeče, jako například Iba analyzer, SPS analyzer, PLC AutoSPy a jiné. Tyto profesionální prostředky disponují širokou škálou možností pro optimální zobrazování dat z průmyslových aplikací. Nevýhodou těchto prostředků však obvykle bývá vysoká cena jednotlivých licencí trendových prohlížečů. Právě vysoká cena za jednotlivé licence může být pro některé případy použití limitující. Tento fakt je jedním z důvodu vzniku myšlenky vytvoření vlastního datového prohlížeče dle vlastních specifikovaných požadavků.

Navržení a vytvoření datového prohlížeče obsahuje širokou problematiku, která představuje například správné načítání dat z různých datových zdrojů, výběr požadovaných signálů, správné formátování dat pro přehledné vykreslení v grafu, tvorbu virtuální signálů, práce s daty, vlastní obsluhu grafického prohlížeče a další. Právě z těchto důvodů obsáhlosti byl vývoj trendového prohlížeče rozdělen do tří diplomových prací. Předložená práce je částí, která se zabývá problematikou vytvoření datového rozhraní. Cílem je vytvořit část zabývající se načítáním a předáváním dat z datových zdrojů do trendového prohlížeče. Na tuto problematiku pak navazuje druhá práce zabývající se problémem zobrazování datových křivek. Poslední práce se pak zabývá problematikou analýzy načtených dat a z naměřených dat pak provádí identifikaci soustav. V obou navazujících pracích je pro přístup k datům využito právě univerzální datové rozhraní, které je tématem této diplomové práce. Tyto tři diplomové práce popisují celkovou problematiku týkající se vytvoření příkladu trendového prohlížeče pro průmyslovou automatizaci.

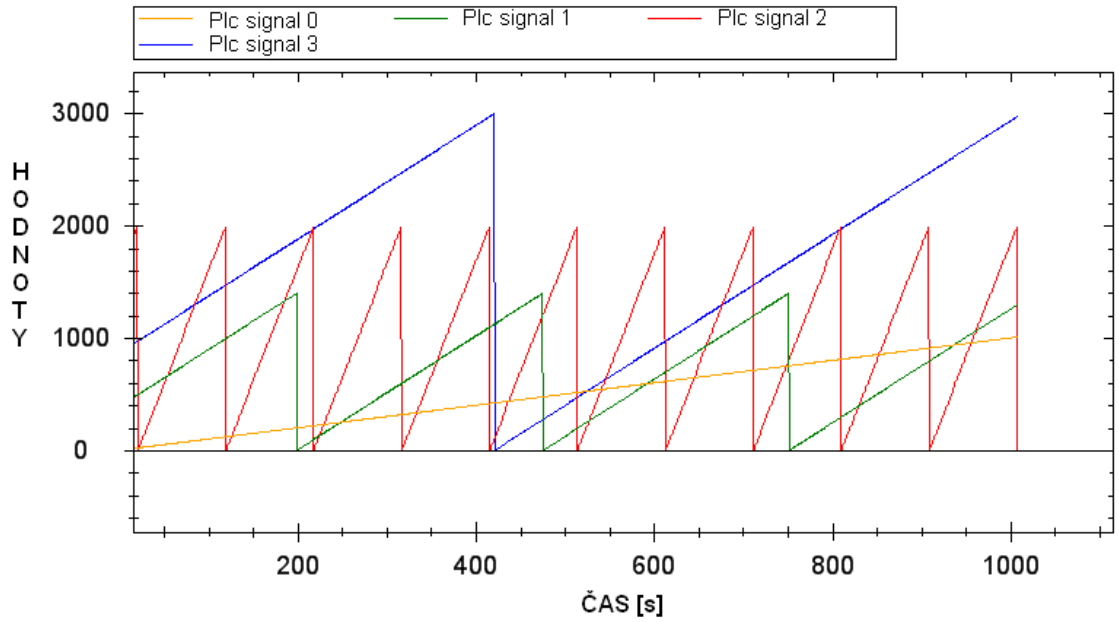
Jak již bylo zmíněno, cílem této diplomové práce je vytvořit datové rozhraní pro datový prohlížeč. Z tohoto důvodu jsou v této práci zhodnoceny metody, požadavky, možnosti datového rozhraní a podle těchto specifikací je proveden návrh a realizace umožňující zprostředkování a přenos dat z datových zdrojů do aplikací navržených za pomocí využití platformy .NET.

V první části práce jsou uvedeny současné nástroje pro prohlížení trendů v průmyslové automatizaci. Následující kapitola uvádí přehled a analýzu dostupných produktů pro přístup ke zdrojovým datům. Po těchto teoretických částech, které popisují současné způsoby čtení dat v průmyslové automatizaci následují části, které se zabývají návrhem a problematikou implementace datového rozhraní za pomoci využití platformy .NET. Mezi poslední kapitoly jsou zařazeny části, které se týkají testování, zhodnocení výsledků a možností vytvořeného datového rozhraní. Jednou nedílnou částí této práce je příloha obsahující rozšířený abstrakt v anglickém jazyce, který sumarizuje celkovou problematiku řešení.

2 Současné možnosti zobrazování dat ve formě trendů

V této kapitole jsou uvedeny současné možnosti, které jsou využívány pro zobrazování dat ve formě trendů. Přehledné zobrazování dat formou trendů patří v průmyslové automatizaci k jednomu z důležitých požadavků. Pod pojmem zobrazování dat pomocí trendů se v průmyslové automatizaci myslí uchovávání hodnot s časovou značkou a grafické znázornění procesních hodnot z PLC na obrazovce PC. Hodnoty mohou v řízeném systému reprezentovat například aktuální velikost tlaku, hladiny, rychlosti nebo teploty. Tyto datové hodnoty se pak v systémech PLC nazývají například proměnné, nebo „tagy“. Při programování systémů PLC je nutné vymezení oblasti paměti pro jednotlivé proměnné. Obecně se může jednat o hodnoty reprezentující vstupní nebo výstupní signály z automatizovaného procesu, nebo také o vnitřní proměnné PLC, sloužící například pro uchovávání mezivýsledků nebo jiných požadavků v programu PLC. Právě načítání hodnot z těchto datových proměnných PLC je základní funkcí datového rozhraní pro trendový prohlížeč.

Tyto požadované hodnoty jsou pak vynášeny do „x,y“ grafů, kde hodnoty osy „x“ jsou data, která ve většině případů nesou informaci o časovém údaji v milisekundách, sekundách, hodinách, dnech a podobně. Také se může jednat o údaj reprezentující aktuální běžící cyklus programu v automatu. Do hodnot osy „y“ jsou pak vkládány hodnoty reprezentující procesní údaj například rychlost, tlak nebo teplotu. Těmito způsoby lze přehledně analyzovat data znázorňující průběhy z regulačních smyček, přechodových dějů, chybových stavů a podobně.



Obr. 1 Příklad obecného trendu vytvořeného pomocí platformy .NET

Trendy v průmyslové automatizaci lze rozdělit do dvou základních skupin. První skupinou jsou „Online trendy“. Využívání online trendů je vhodné například při nastavování důležitých parametrů systému, které mají vliv na okamžitou změnu chování v řízeném systému a díky názornému vykreslování dat formou křivek pak napomáhají přehledně znázornit aktuální informace o chování řídicího systému, na které pak může operátor nebo programátor rychle reagovat. V trendech se do grafů vykreslují data z řídicího systému znázorňující aktuální

hodnoty z požadovaných signálů. Konkrétně se může jednat například o data ze systému PLC Simatic S7 a podobně. Následně se tato data z řídicích systémů aktualizují do grafu trendů dle zvolených aktualizačních intervalů. Jsou definovány v řádu milisekund až minut či hodin. Nastavení intervalů je důležité optimalizovat, jak z hlediska dynamiky řízeného systému z důvodu možnosti výskytu zkreslení zobrazovaného signálu, tak z hlediska objemu přenášených dat.

Druhým případem jsou Offline trendy, které slouží pro zobrazení historických dat z řídicího systému uložených například v souboru nebo v databázi. Může se jednat například o záznamy dat trvajících v řádech sekund až minut hodin, popřípadě dnů nebo měsíců či let. Při zobrazování těchto dat je důležité znát informace o formátování uložených záznamů. Zde je uveden výčet nejznámějších typů: CSV soubor, Binární soubor, ODBC, XLS.

Na současném trhu existuje mnoho softwarových produktů, kterými lze pomocí trendů zobrazovat data z řídicích systémů. Tyto produkty lze rozdělit do dvou základních skupin a to na data analyzéry, jejichž hlavním úkolem je především zobrazování a ukládání procesních dat z řídicího systému. U těchto prostředků je nutné, aby umožňovaly znázorňovat data načítána z každé periody cyklu programu PLC, což zaručí znázornit všechny vzorky dat z regulačních smyček nebo náhodné chybové špičkové hodnoty.

Dále lze pak data formou trendů znázornit za pomocí vizualizačních softwarů, které umožňují celkovou obsluhu řízeného procesu a mimo to také obsahují trendové nástroje pro znázornění dat formou trendů. U tohoto typu znázorňování dat však nelze očekávat přesnost dat z každého cyklu PLC. Vzorkovací frekvence vzorků se pohybuje okolo 100 ms. Perioda vzorků dat je závislá na použitých komunikačních prostředcích, jako například OPC.

V této kapitole jsou dále rozebrány některé dostupné možnosti zobrazování dat z řídicích systémů formou trendů za pomocí využití některých produktů pojednávajících o skupině analyzérů, v dalších částech je pak zmíněno o možnostech zobrazování dat pomocí vizualizačních softwarů.

2.1 PLC-ANALYZER pro 5

Jedná se o softwarový systém od firmy Autem [4], který nabízí mnoho způsobů pro získávání a vyhodnocování dat z automatizovaných systémů, které jsou ovládány pomocí PLC. Software umožňuje prohlížení offline i online dat. Umožňuje zobrazování a vyhodnocování požadovaných datových proměnných, vstupních a výstupních signálů z řídicího systému. Online zobrazení umožňuje pozorování časového průběhu zvolených signálů v reálném čase s přesností dat na jeden cyklus PLC. Tato možnost je důležitá proto, aby bylo možno získat informace ze všech vzorků dat, která jsou v řídicím systému zpracovávaná. Cyklus PLC je celek programu, který se periodicky provádí v PLC. Obecně se skládá z několika částí, načtení vstupních signálů, zpracování programu, zápisu výstupních signálů. Čas trvání neboli perioda jednoho cyklu nemusí být vždy stejně dlouhá. Může být závislá na vstupních hodnotách, které vstupují do matematických operací na vykonávání různé délky programu, která může záviset na vstupních podmínkách.

Z hlediska záznamu je tento softwarový nástroj PLC-ANALYZER pro 5 schopen analyzovat data průběžného záznamu a také data, která jsou vyvolána náhlou událostí. K takovémuto způsobu záznamu používá systém zvolené spouštěcí podmínky neboli „triggery“. Díky této

metodě je analyzer schopen zachytit a analyzovat stavy, které se v řídicím systému vyskytují jen velmi náhodně, především poruchové stavy.

Jednou z hlavních výhod tohoto systému je to, že pro získávání dat z PLC je využito rozhraní, která jsou v systémech PLC obecně podporována. Popisovaný analyzer podporuje čtení dat z řídicího systému za pomocí využití standardních automatizačních sítí jako například Profibus, Ethernet, MPI/PPI. Jako další, rozšiřující, funkcí analyzeru je možnost použít mečící adaptér „AD_USB-Box“, což umožní zaznamenávat externí napěťové a proudové technologické signály, které nejsou k dispozici z PLC. Pro dálkovou údržbu je schopen analyzer pracovat také přes veřejnou telefonní síť za pomoci zařízení „terraLink“. [3] [4]

Shrnutí hlavních vlastností

- Logická analýza
- Systémová analýza
- Diagnostika poruchových stavů
- Měření požadovaných hodnot
- Porovnávaní souborů naměřených dat
- Snímání dat s přesností na jeden cyklus PLC
- Simultánní snímaní dat z několika PLC systémů
- Při připojeni PLC k PC není potřeba programových ani hardwarových úprav spojení na straně PLC
- Možnost automatických alarmů v případě výskytu spouštěcích událostí, možnost zaslání SMS, E-mailů
- Vícejazyčná podpora
- Možnost záznamu externích analogových signálů pomocí zařízení AD_USB-Box
- Export dat do formátu HTML a CSV - soubor
- Import externích dat ve formátu CSV - soubor

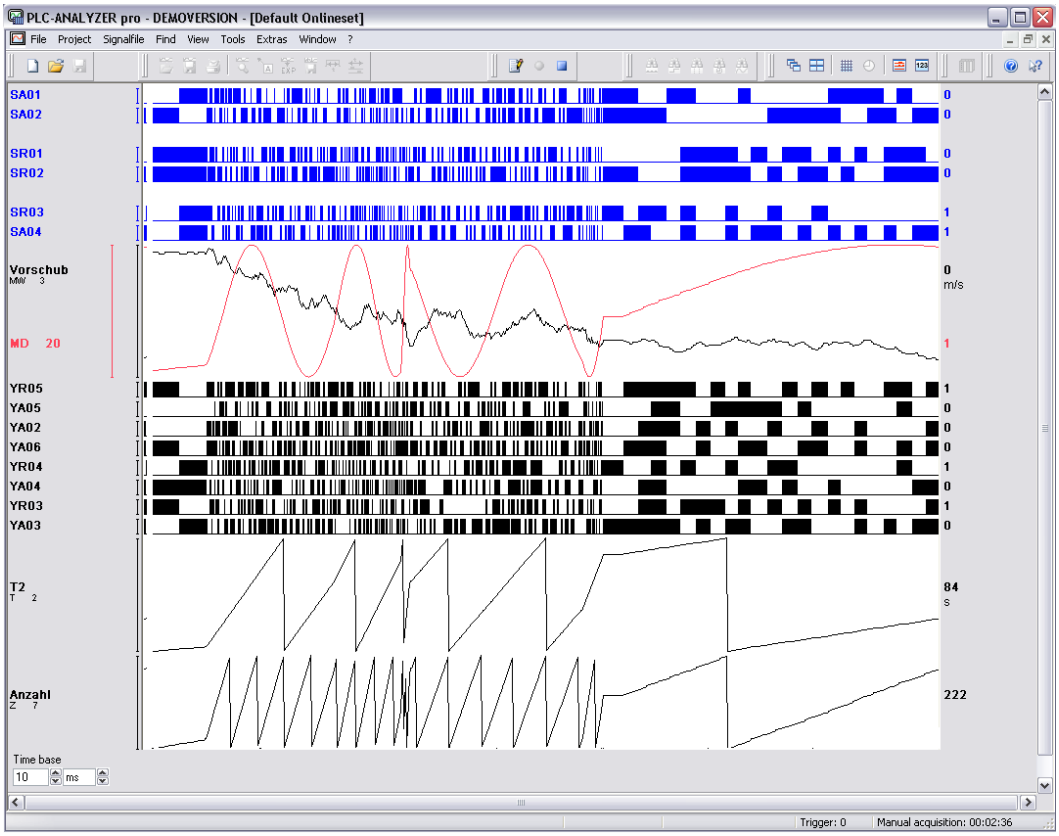
Oblast použití

- Diagnostika poruch v PLC
- Lokalizace a diagnostika náhodných chyb
- Dlouhodobý záznam hodnot
- Vývoj a servis PLC zařízení
- Dokumentace dat pro systém řízení jakosti

Komunikační možnosti

Využití automatizačních síťových nebo programovacích rozhraní dle typu zvoleného PLC. Podporované drivery jsou například tyto:

- Siemens SIMATIC S7***	Profibus, Ethernet TCP/IP , MPI/PPI	
- Siemens SIMATIC S5	Ethernet TCP/IP	
- Siemens LOGO!	Programming interface	
- Siemens SINUMERING (S5)	Programming interface	
- Siemens SIMOTION C/P/D	Profibus, Ethernet TCP/IP , MPI/PPI	
- PHOENIX ILC	Ethernet TCP/IP	
- B&R	Ethernet TCP/IP	
- MITSUBICHI MELSEC Q/A	Ethernet TCP/IP	[3] [4]



Obr. 2 Obrazovka PLC - Analyzer pro 5[4] (testování ukázková demoverze)

2.2 AutoSPy

Analyzátor AutoSPy je softwarový produkt německé firmy. Jedná se o výkonný softwarový nástroj, který slouží pro optimalizaci a hledání poruch v průmyslových procesech. Softwarový nástroj se skládá z hlavní aplikace, která slouží pro vizualizaci, archivaci a vyhodnocování načítaných dat, která jsou z požadovaných zdrojů dostupná za pomoci využití Plug - in modulů. Tyto moduly neboli ovladače slouží pro sběr dat z PLC či vizualizace. Analyzer AutoSPy je určen především pro čtení dat z automatů Siemens, a proto je jedním z dostupných modulů ovladač pro PLC Siemens - Simatic S5 a S7. Za pomoci tohoto ovladače umožňuje také tento

analyzátor, stejně jako první popisovaný PLC-ANALYZER pro 5, používat některé výhodné metody, jako je například nepotřebná úprava PLC programu před zahájením čtení dat do PC, či zaznamenávání dat z několika PLC současně do jednoho vyhodnocovacího protokolu, nebo sbírání a vyhodnocování dat s přesností na jeden cyklus PLC, popřípadě s přesností na okamžik vzorkování. Mezi jednu z výhod tohoto analyzeru je umožnění uživateli vytvářet uživatelské značky v signálových stopách. Tyto značky mohou posloužit pro měření některých důležitých parametrů signálu jako například měření špiček signálů, nebo časový rozkmit. Za zmínku stojí také to, že program analyzeru zůstává nadále obsluhovatelný i v případě vypadnutí komunikace s PLC. Archivace naměřených dat je možno provádět pomocí uložení projektu s informacemi o měření, nebo jej lze tak jako v případě PLC-ANALYZER pro 5 exportovat do CSV souboru. [5]

Shrnutí hlavních vlastností

- Systém umožňuje záznam z více datových zdrojů do jednoho protokolu
- Hlavní aplikaci lze pomocí Plug-in modulu rozšiřovat o přístupy k datům z různých datových zdrojů
- Možnost libovolného uživatelského značení pomocí značek ve stopě trendů
- Jednotlivá okna v aplikaci lze volitelně uživatelsky rozmisťovat
- Ukazování aktuálních hodnot signálů při pohybu kurzorem

Oblast použití

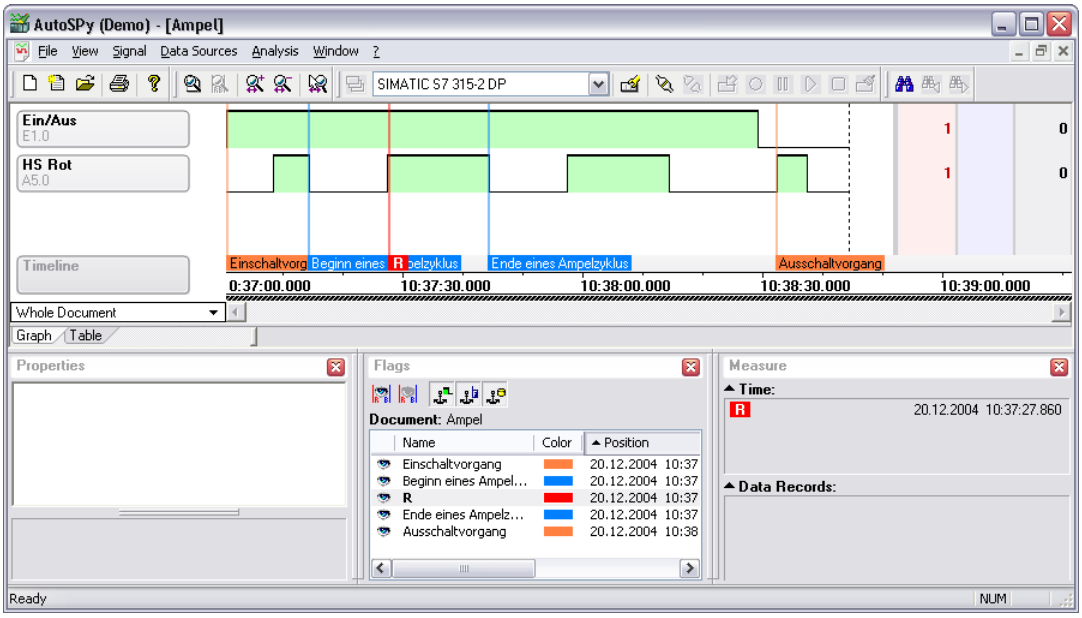
- Diagnostika poruch v PLC
- Lokalizace a diagnostika náhodných chyb
- Dlouhodobý záznam hodnot
- Vývoj a servis PLC zařízení

Komunikační možnosti

Podporované komunikační možnosti závisí na dostupnosti aktuálního Plug - in driveru. Může se jednat například o tyto způsoby komunikace:

- Převodníky ACCON NetLink, NetLink PRO, NetLink USB
- Karta ACCON PROFIBUS/MPI
- TCP/IP komunikace pro SIMATIC řady S7-300 CP343-1, S7-400 CP443-1
- Karty PROFIBUS od společnosti Softing

[5]



Obr. 3 Obrazovka Auto SPy [5] (testování ukázková demoverze)

V dostupné demoverzi lze konfigurovat data pouze pro dva signály v základu z těchto datových zdrojů: LabJack U12, OPC DataAcces, SIMATIC S5,SIMATIC S7-300/400.

2.3 ComesHistorian

Tento analyzer od české společnosti Compas automatizace, s názvem ComesHistorian je jednou z částí několika modulů z informačního modulárního systému Comes. Popisovaný modul se skládá z uživatelské a konfigurační části, lze jej samostatně využívat pro sběr a archivaci dat z řídicích systémů, za účelem tvorby trendů a archivaci procesních dat. Tyto získávána data pak lze zprostředkovávat dalším modulům informačního systému Comes, mezi které patří například Comes tracebility, což je produkt pro sledování například použitých surovin a obalů (v potravinářském odvětví) nebo mohou být tato data distribuována modulu Comes modeler., což je nástroj sloužící pro přetváření a upravovaní dat. Mimo těchto možností předávání dat do dalších modulů lze data prezentovat za pomoci využití webových klientů v podobě grafů pro analogové i digitální veličiny.

Tak jako předchozí zmiňované analyzery umožňuje i tento software zálohovat data, konfiguraci, import a export dat například do datového formátu XLS, XML, CSV, TXT. Systém ComesHistorian pak dále umožňuje provádět základní statistiky, porovnávání jednotlivých trendů pomocí možnosti funkce využití více časových os nebo provádění analýzy uložených dat s možnostmi hlášení. [6]

Shrnutí hlavních vlastností

- Možností porovnávaní různých datových trendů s možnostmi více časových os
- Archivace hlášení řídicích systémů a jejich zpracování
- Podpora kapacitního plánování výroby
- Exporty dat do formátu XLS, XML, CSV, TXT

Oblast použití

- Všeobecné použití v řídicích systémech, ve kterých jsou kladeny vysoké požadavky na veliké objemy ukládaných dat a celkovou spolehlivost systému, například potravinářský průmysl

Komunikační možnosti

- Pomocí standardizovaného rozhraní OLE DB

[6]

Obr. 4 Obrazovka Comes Historian [7]

2.4 Trendové nástroje vizualizací

Trendové nástroje v Intouch SCADA HMI

Jedná se o vizualizační software od firmy Wonderware, který slouží pro vizualizaci a řízení supervizích výrobních technologií a procesů. Vizualizace umožňuje technologům a operátorům přehledné znázornění stavů v technologickém procesu pomocí široké škály grafických nástrojů. Znázornění dat z technologických signálů je ve vizualizaci možno provádět pomocí grafických číselníků nebo je lze zaznamenávat pomocí nástrojů pro tvorbu historických a reálných trendů. Pro odchycení výjimečných stavů, lze ve vizualizačním prostředí využívat širokou škálu alarmních hlášení. Vizualizační systém InTouch umožňuje získávat data pomocí technologií OPC serveru, DDE, NetDDE a Wonderware SuiteLink. [8]

Obr. 5 Ukázka vizualizačního okna Intouch [8] s trendovými možnostmi

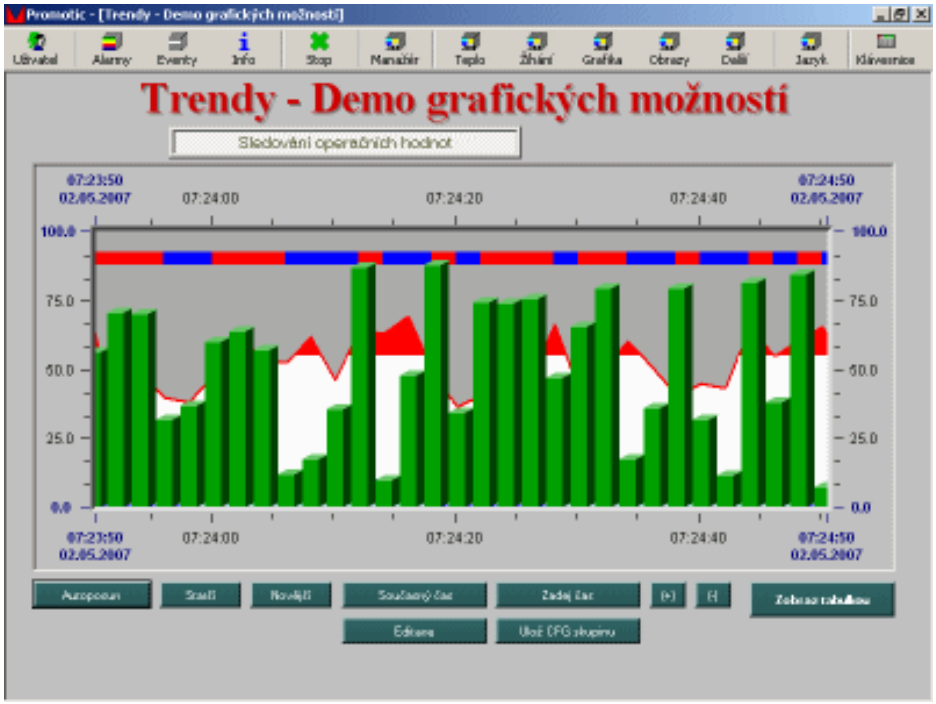
Trendové nástroje v WinCC

Jedná se o vizualizační software pracující na platformě Windows, který je určen pro všechny náročné průmyslové aplikace. Sledování a provádění operátorských zásahů lze provádět pomocí Intranet / Internet rozhraní. Systém WinCC výrazně podporuje spojení k vlastním průmyslovým automatům řady Simatic, dále pak umožňuje navázání na relační databázi MS SQL server, které pak ukládá konfigurační i archivační data. Pomocí této koncepce lze k požadovaným datům přistupovat pomocí metod ODBC, SQL. Pro aplikace, které jsou spuštěny paralelně, například MS Excel je umožněna spolupráce na datech pomocí DDE. Systém WinCC umožňuje znázorňování aktuálních informací o řídicím systému za pomocí široké škály grafických prvků, jako například virtuálních displejů, budíků datových tabulek a podobně. Mimo tyto prvky lze ve WinCC znázorňovat datové aktuální i archivované veličiny formou trendů. K tomuto účelu slouží objekty TrendControl, pomocí kterého lze nastavovat samostatné osy pro jednotlivé signály a exportovat vybraná data například do CSV souboru. [9]

Trendové nástroje v Promotic

Systém Promotic je komplexní softwarový objektově orientovaný SCADA nástroj, který slouží tak, jako předchozí popisované vizualizační nástroje k monitorování a operátorským zásahům v různých technologických řídicích procesech. Tento systém je určen pro operační systémy Windows 8/7/Vista/XP/XPe/2003-8Server.

Vizualizační systém Promotic obsahuje zabudované rozhraní OPC, XML, DDE, dále pak nabízí pro tvorbu grafického rozhraní širokou škálu vizualizačních nástrojů včetně těch, které slouží k tvorbě trendů. Trendové hodnoty lze znázorňovat buď graficky, nebo tabulkově. Vybrané signály lze ukládat například ve formátech DBase, Acces, MySQL. [10]



Obr. 6 Ukázka vizualizačního okna Promotic [10] s trendovými možnostmi

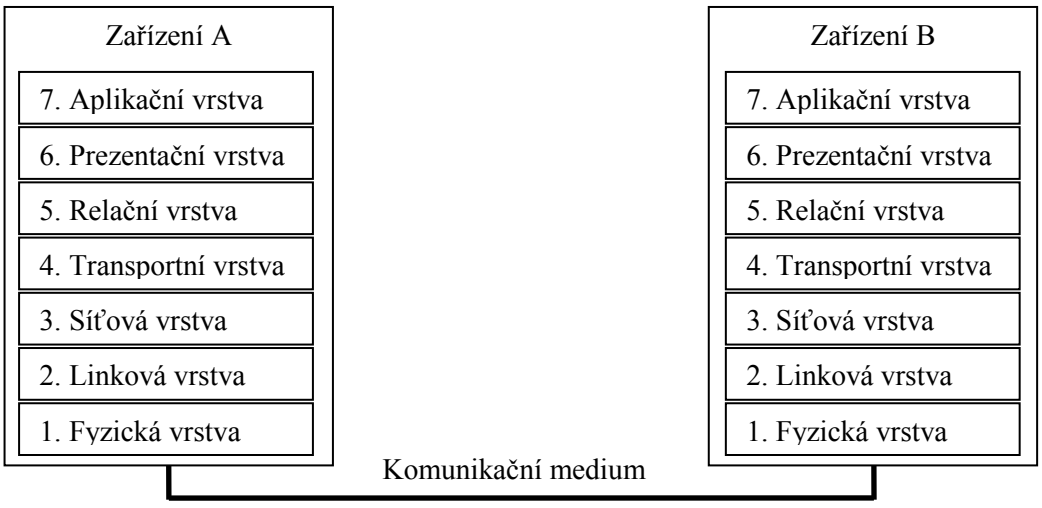
3 Možnosti přístupů k datům

Tato kapitola pojednává o způsobech a možnostech sloužících pro přístup k požadovaným datům z řídicích systémů. V předchozí kapitole zabývající se zobrazováním dat z řídicích systémů formou trendů je zmínka o možných komunikačních způsobech popisovaných trendových prohlížečů. Právě komunikační způsoby celkově ovlivňují možnosti přenosu dat například z hlediska rychlosti vzorkování nebo přenášeného objemu informací. Výběr způsobů komunikace závisí na daných možnostech komunikujících zařízení a to, jak z hlediska hardwarových, tak softwarových možností. Z hardwarového kritéria omezují zařízení možnosti fyzického komunikačního interfacu neboli fyzické komunikační vrstvy, například komunikace pomocí sítě Ethernet, Profibus, RS485 a podobně. Tato kritéria jsou pevně stanovena výrobcem zařízení a lze jej v případě možností doplnit přídavnými komunikačními kartami.

V případě, že zvolené zařízení podporuje požadované fyzické komunikační možnosti, je pak dále pro vytvoření komunikace potřeba zvolit požadovaný komunikační protokol a prostředky pro předání dat až do podoby konečného uložení v databázi, nebo znázornění v trendovém prohlížeči. Popis bezproblémového způsobu přenosu dat mezi zařízeními je názorně popisován pomocí standardizace ISO/OSI modelu, který byl vydaný organizací ISO v roce 1983. Rozdělení problematiky předávání dat mezi zařízeními v počítačové síti je rozděleno ISO/OSI do sedmi vrstev, které jsou blíže popsány v prvních částech této kapitoly. Následující částí této kapitoly pak popisují konkrétní prostředky pro přístup k zdrojovým datům. [12]

3.1 Popis ISO/OSI modelu

Jak již bylo zmíněno, ISO/OSI model se skládá ze sedmi vrstev. Zde jsou části jednotlivých vrstev blíže popsány.



Obr. 7 Znázornění vrstev ISO/OSI modelu [12]

Komunikační medium - je závislé na volbě typu sítě, může se jednat například o kabelové, bezdrátové, nebo optické vedení.

Fyzická vrstva

Jedná se o hardwarovou vrstvu popisující parametry přenosového média z hlediska elektrických, mechanických, popřípadě bezdrátových vlastností. Vrstva definuje fyzické prostředky, kterými dochází k výměně datových informací mezi komunikujícími zařízeními, konkrétní hardware komunikačního zařízení.

Linková vrstva

Jedná se o vrstvu, která se někdy nazývá spojová vrstva nebo vrstva síťových rozhraní. Vrstva slouží k formátování bloků přenášených dat, jejich detekci chyb, přístupu na fyzické médium, synchronizace a řízení toku dat.

Síťová vrstva

Tato vrstva je důležitá především v případě, že se nejedná o přímé spojení mezi uzly, ale o spojení probíhající přes jeden, nebo více spojových uzlů. V tomto případě je pak nutno data z jednotlivých uzlů směrovat na požadované koncové uzly pomocí síťových směrovacích zařízení. Tato vrstva vyhledává dostupnou cestu dat mezi jednotlivými uzly dle možností použité topologie sítě.

Transportní vrstva

Jedná se o vrstvu definující „end to end“ spojení systémů. Jelikož předchozí síťová vrstva zajistí vyšším vrstvám představu vytvoření přímého síťového propojení, tak tato vrstva definuje pouze spojové protokoly pro strukturované zprávy, dále pak rozděluje různě veliká vysílána data do jednotlivých paketů definovaných velikostí, a díky čemuž pak není omezena velikost přenášených dat.

Relační vrstva

Jedná se o vrstvu, která slouží k vytváření spojení mezi jednotlivými účastníky, dále pak koordinuje komunikaci a udržuje vytvořené spojení po stanovenou dobu.

Prezentační vrstva

Úkolem této vrstvy je vytvářet datové konverze, komprese, dekomprese popřípadě šifrování dat. Celkově tato vrstva zabezpečuje řešení rozdílů mezi formátování dat s aplikací a síťového rozhraní.

Aplikační vrstva

Jedná se o nejvyšší síťovou vrstvu, která reprezentuje způsoby definující přístup koncových aplikací do sítí. Ke konečnému přístupu k fyzickému médiu sítě jsou využívány všechny nižší vrstvy ISO/OSI modelu. [11] [12]

3.2 Síťová komunikace Profibus

Jedná se o sběrnici, která je určena pro širokou oblast automatizačního průmyslu, jako například automatizace výrobních linek, řízení výroby. Sběrnice Profibus je z důvodu standardizace popisována evropskou normou EN50170. Z hlediska optimalizace rychlosti odezvy je tato sběrnice popsána pouze na třech vrstvách ISO/OSI modelu, a to na fyzické, linkové a aplikační. Pro komunikaci na fyzické vrstvě lze využívat metalické vedení komunikačního standartu sériové komunikace RS - 485 a to jak standardní provedení, tak verzi proudové smyčky dle normy IEC 1158-2. Další možností je využití optického vlákna. Přenosová rychlost sběrnice je se odvíjí od délky a typu použitého komunikačního média, pohybuje se v rozmezí od 9 kbit/s až 12Mbit/s. Pro řízení komunikace na sběrnici jsou využívány metody, „token pasing“, „token ring“, „master – slave“, „klient - server“.

Komunikační sběrnice se rozděluje na tři standardy, které jsou využívány podle oblasti použití Profibus sběrnice, jedná se o následující rozdělení:

Profibus DP

Jedná se o nejčastější formy využití této sběrnice. Tento typ využívá komunikaci typu master - slave, používá se především na úrovních nižších vrstev systémových úrovní distribuovaných systémů, čímž je myšleno například čtení signálu do automatu se vzdálených vstupně výstupních zařízení z Profibus rozhraním. U tohoto způsobu komunikace je kladen důraz na rychlost odezvy dat. Pro fyzikou komunikaci je využíváno možností RS 485, nebo optického vlákna.

Profibus FMS

Tento typ komunikace se využívá v různorodém síťovém prostředí. Tímto je myšleno, že jsou v komunikaci zařazeny různorodé systémy, jako například PLC, Operátorské panely, PC. V tomto typu komunikace není kladen důraz na rychlosti odezvy, ale na možnosti sítě vytvořit širokou síť z možností spojení zařízení v mnoha síťových vrstvách od mnoha výrobců. Nyní se tato verze sběrnice nahrazuje ethernetovými komunikačními možnostmi, avšak v případě využití Profibusu FMS je pro přenos informací využito fyzické komunikace na bázi RS 485, nebo optické vlákno.

Profibus PA

Jedná se o rozšířenou normu Profibus DP. Tato verze je určena pro sběr dat ze snímačů s Profibus rozhraním, které jsou umístěny v nebezpečném prostředí s možností nebezpečí výbuchu. Z tohoto důvodu je využito komunikačního standardu proudové smyčky IEC 1158-2.

[12] [13] [14]

3.3 Síťová komunikace Ethernet

Ethernetová síť je rozsáhlou problematikou, tato práce se zabývá tématem, které tuto síť využívá pro přístup k datům, a proto jsou zde uvedeny a popsány jen základní informace o síťové komunikaci na bázi Ethernetu.

Ethernet

Síťová komunikace Ethernet je v současné době nejrozšířenějším typem komunikace. Je založena na principu nazývaném CSMA/CD. Princip CSMA říká, že stanice musí provést před zahájením vysílání ověření, zda komunikační médium nepoužívá jiná stanice. Metoda CD má za úkol zpětně kontrolovat, zda je na médium signál o logické úrovni, která odpovídá vysílanému signálu. Takový případ může nastat například v případě, že by nedopatřením nastal zkrat na síti nebo by bylo provedeno neoprávněné vysílání jiného zařízení. V případě detekce této chyby neboli kolize pak jednotlivé stanice vygenerují náhodný čas, po kterém pak nastane nové opakování přístupu na síť. Četnost těchto možných kolizí je ovlivňována počtem zařízení, které jsou připojovány k síti. Pro komunikaci používá Ethernet komunikační média, podle kterých se pak odvíjí možnosti přenosových rychlostí. Tyto komunikační rychlosti se pohybují od 10 do 1000 Mbit/s. Předávání dat je prováděno pomocí paketu neboli bloku přenášených dat, který je pro všechny modifikace Ethernetu obdobný. Pro směrování těchto paketů jsou v komunikaci pomocí Ethernetu využívány MAC adresy a IP adresy. [15]

Profinet

Tato komunikační síť vychází ze standardů sítě Ethernet, avšak disponuje některými zásadními změnami. Tato síť je určená pro využití v průmyslových aplikacích, ve kterých je kladen důraz na spolehlivost a rychlou odezvu aktuálních dat. Jak již bylo zmíněno, tak v ethernetové komunikaci může docházet ke vzniku kolizních stavů zařízení, které mohou být závislé na počtu těchto připojených zařízení k sběrnici. S tohoto důvodu je sběrnice Profinet koncipována tak, že je síť fyzicky rozdělávána podle funkčnosti a významností zařízení do jednotlivých bloků sítě. [16]

Ethernet Power link

Jedná se o komunikaci určenou pro řídicí systémy s důrazem na řízení v reálném čase. Tento standard je postaven na vlastnostech sítě Ethernet. Odlišnost v této síti spočívá v důsledném rozdělení časového cyklu na dvě části. Jedná se o část, která je určena pro přenášení časově kritických dat. Další část slouží pro předávání časově nekritických dat. [17]

3.4 Komunikační standardy pro přístup aplikací k zdrojovým datům

V této podkapitole jsou uvedeny možnosti, které mohou aplikace využívat pro přístup k datovým zdrojům z řídicích systémů nebo jiných dostupných uložit v rámci PC.

DDE

Tento komunikační prostředek slouží k dynamické výměně dat mezi různými aplikacemi v rámci PC. Příkladem aplikací, mezi kterými dochází k výměně dat je například Excel, vizualizační, trendové nebo databázové nástroje. Komunikace dat probíhá tak, že se vytvoří DDE server, ve kterém jsou pak uloženy požadované datové body určené pro načítání klientem. Dle nastavení serveru pak mohou klienti z těchto datových bodů číst, nebo do nich zapisovat datové hodnoty. [18]

NetDDE

Tento typ je rozšířením standardu DDE o komunikaci v síťovém rozhraní. Pomocí NetDEE lze přenášet data v rámci lokální sítě. Tak jako základní DDE tak obsahuje i tento rozšířený komunikační standart určité parametry pro tvorbu nastavení spojení. [19]

OPC

Zkratka OPC (Object Linking and Embeding for Process Control) je označení pro komunikační protokol, který má za úkol vytvořit jednotné komunikační rozhraní pro výměnu dat mezi softwarovými produkty a hardwarem určeným pro procesní řízení. Za pomocí využití komunikačního protokolu pak není potřeba při návrhu koncové aplikace využívající data zvlášť definovat komunikační ovladače pro připojení k řídicím systémům. Při návrhu stačí pouze implementovat rozhraní pro připojení k OPC protokolu. Výrobci průmyslových zařízení pak dodržují standardy komunikace OPC protokolu. Díky tomuto standardu pak OPC protokol slouží jako ovládač zprostředkující spojení s konečnou softwarovou aplikací a průmyslovým procesním zařízením. Přesný standard OPC rozhraní je definován organizací OPC Foundation. Díky využití tohoto standardu pak konečnou vytvořenou aplikaci pro práci s daty například nezajímá, zda získává data ze zařízení Siemens, nebo Mitsubichi či jiných, mimo to je také pro konečnou aplikaci nepodstatná specifikace komunikační sběrnice. Všechny tyto problémy jsou řešeny v softwarovém mezičlánku OPC.

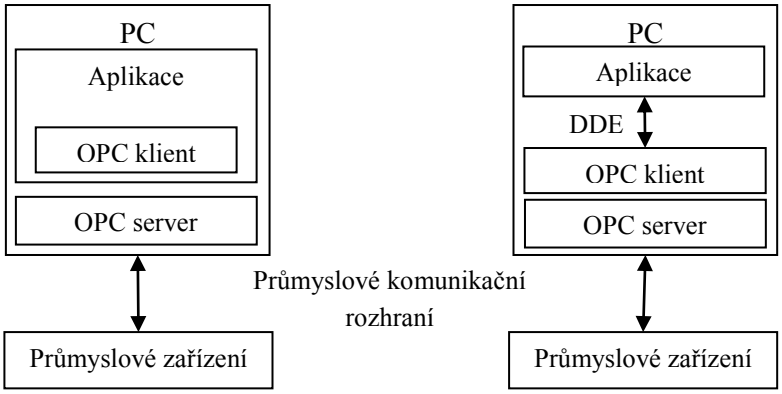
Komunikační protokol OPC pracuje na principu klient - server. Pro vytvoření spojení mezi hardwarem a softwarem za pomocí OPC protokolu musí být vždy spuštěny dva typy OPC softwaru, OPC klient a OPC server.

OPC server je označení pro software přijímající data z připojených zařízení sloužící pro řízení v automatizaci. Tato data jsou přijímána za pomoci využití standardizovaných průmyslových komunikačních prostředků, jako třeba například Profibus, Siemens MPI, PPI, Modbus a podobně. Data získávána ze zařízení pak OPC server dále poskytuje OPC klientovi ve formátu OPC protokolu.

OPC klient je označení pro software přijímající data formátu OPC protokolu z OPC serveru. Data jsou pak dále v požadovaném formátu poskytována koncovým aplikacím. Jako například vizualizace databázové systémy a podobně. [20]

V praxi lze vytvořit komunikační spojení za pomocí využití OPC nejčastěji následujícími způsoby:

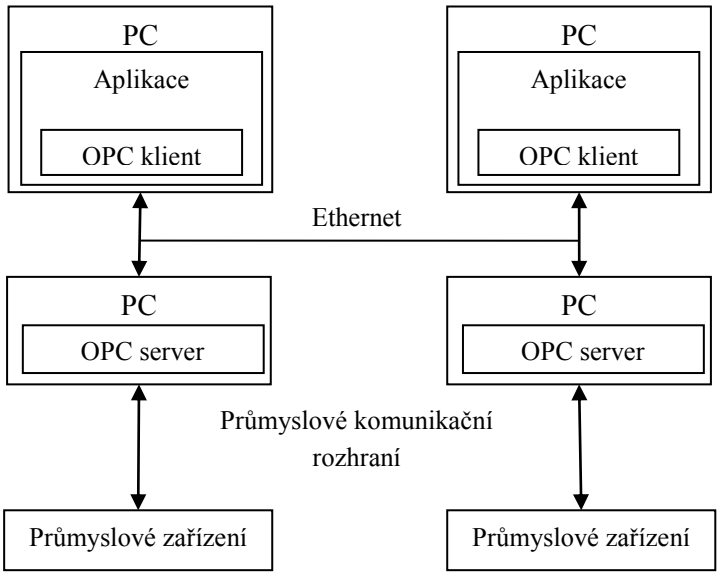
1. OPC klient a server jsou zpuštěny současně s potřebnou aplikací na jednom PC.



Obr. 8 OPC server a klient v rámci jednoho PC

Na Obr. 8 OPC server a klient v rámci jednoho PC je znázorněno, že data jsou z průmyslového zařízení do aplikace dodávána pomocí OPC serveru a OPC klienta, který může být součástí aplikace, nebo se může jednat o samostatnou implementaci OPC klienta. V takovém případě lze použit mezi aplikací a OPC klientem spojení pomocí využití komunikačního standardu DDE.

2. OPC server je spuštěn na děleném počítači od OPC klienta, komunikace mezi OPC klientem a serverem probíhá pomocí ethernetové sítě. OPC klientů a serverů může být v rámci jedné sítě zapojeno několik.



Obr. 9 Spojení několika OPC serverů a klientů

OPC server umožňuje poskytovat data několika klientům a pro ty platí možnost přijímat data z více serverů.

Zhodnocení výhod a nevýhody komunikace za pomocí využití OPC a DDE

Výhody: OPC a DDE prostředky umožňují získávat data z široké oblasti PLC systému. Existuje mnoho OPC prostředků, které umožňují navázat bezproblémové zprostředkování dat do vizualizací a databází a podobně. Vždy je jen potřeba použít patřičné systémy pro daný typ zařízení. Jelikož OPC a DDE je standardizován, tak je možno získávat data z široké škály průmyslových zařízení do mnoha typů PC aplikací a databázových systémů. Ve většině případů je tato komunikační metoda OPC výhodná a dostačující.

Nevýhody: OPC standart negarantuje poskytování dat z každého cyklu PLC programu. Z tohoto důvodu není možné využívat OPC standart pro přesnou analýzu rychlých regulačních smyček nebo znázornění náhodných špičkových hodnot.

3.5 Komunikační produkty OPC

Komunikační OPC servery jsou dostupné pro většinu výrobců průmyslových automatů. Zde je pro příklad uveden seznam dostupných OPC serveru pro jednotlivé typy výrobců:

OPC servery Matricon

Allen Bradley OPC Server Suite
GE Fanuc PLC OPC Server
Honeywell HC900 OPC Server
Mitsubishi PLC OPC Server for Mitsubishi PLCs
Modbus OPC Server for Modbus Devices
OMRON OPC Server for OMRON PLCs
Siemens S7 PLC OPC Server for Siemens PLCs
Siemens SIMATIC TI505 OPC Server
MatrikonOPC Genie (Generic Information Exchange) [23]

Graybox

Graybox OPC Server Toolkit V3
Graybox OPC DA Auto Wrapper [24]

QuickOPC

QuickOPC.NET
QuickOPC-COM
QuickOPC-UA [25]

Seznam nabízených OPC serverů firmou Merz s.r.o.

Informativně jsou u jednotlivých produktů uvedeny ceny ke dni 29.4.2013.

OPC Server HDA ODBC	19 990 Kč
OPC Server pro ADAM 4000	2 990 Kč
OPC Server pro Allen Bradley ControlLogix	23 990 Kč
OPC Server pro Allen Bradley DF1	2 990 Kč
OPC Server pro Allen Bradley Ethernet	23 990 Kč
OPC Server pro Allen Bradley Unsolicited Ethernet	23 990 Kč

OPC Server pro Allen-Bradley DH+	23 990 Kč
OPC Server pro AMiT DB-Net/IP	4 900 Kč
OPC Server pro AutomationDirect - Koyo DirectNET	23 990 Kč
OPC Server pro AutomationDirect - Koyo K Sequence	23 990 Kč
OPC Server pro AutomationDirect EBC	23 990 Kč
OPC Server pro AutomationDirect ECOM	23 990 Kč
OPC Server pro Fatek Facon	2 990 Kč
OPC Server pro GE CCM	23 990 Kč
OPC Server pro GE Ethernet	23 990 Kč
OPC Server pro GE Focas 1 Ethernet	23 990 Kč
OPC Server pro GE SNP	23 990 Kč
OPC Server pro GE SNPX	23 990 Kč
OPC Server pro inkubátory Binder	2 990 Kč
OPC Server pro Koyo DirectNet	2 900 Kč
OPC Server pro M-Bus	2 990 Kč
OPC Server pro Modbus/JBus Slave RTU	2 990 Kč
OPC Server pro MET čítače částic	2 990 Kč
OPC Server pro Mitsubishi Ethernet	23 990 Kč
OPC Server pro Mitsubishi FX	23 990 Kč
OPC Server pro Mitsubishi FX	2 990 Kč
OPC Server pro Mitsubishi FX Net	23 990 Kč
OPC Server pro Mitsubishi Serial	23 990 Kč
OPC Server pro Modbus/Jbus Master RTU	13 300 Kč
OPC Server pro Modbus/Jbus Master RTU Windows Vista, pro Windows 7	13 300 Kč
OPC Server pro SAIA S-Bus	13 300 Kč
OPC Server pro Siemens Simatic 505 Ethernet	23 990 Kč
OPC Server pro Siemens Simatic 505 Serial	23 990 Kč
OPC Server pro Siemens Simatic MPI	8 900 Kč
OPC Server pro Siemens Simatic MPI EC1	8 900 Kč
OPC Server pro Siemens Simatic RK512	13 300 Kč
OPC Server pro Siemens Simatic S5 AS511	6 800 Kč
OPC Server pro Siemens Simatic S7 200/300/400 TCP/IP Ethernet	23 990 Kč
OPC Server pro Siemens Simatic S7-200	23 990 Kč
OPC Server Value Keeper	8 900 Kč
OPC to OPC - OPC koncentrátor	13 300 Kč

[21]

3.6 Komunikační prostředek firmy Ingeteam a.s.

Jedním z možných komunikačních prostředků, který lze použít pro přístup k datům z PLC je prostředek dostupný z firmy Ingeteam a.s. Jedná se o možnost vyčítání dat z PLC Siemens řady Simatic S7. Zmiňovaný komunikační způsob byl původně navržen a aplikován pro načítání dat z PLC pro následné ukládání do databázového systému. Výhoda tohoto komunikačního způsobu spočívá v možnosti poskytování dat z každého periodicky volaného cyklu PLC s pevnou nebo

proměnnou dobou periody. Díky této metodě nemůže dojít ke ztrátě žádných datových vzorků, a proto lze získat informace o všech datech, která se v PLC zpracovávají. Což je zvláště důležité při testování regulačních smyček, kdy je potřeba znázorňovat všechna data s přesnou vzorkovací frekvencí.

Výměna dat mezi PLC a PC probíhá za pomocí ethernetového rozhraní. Pro možnost využití této komunikační metody je potřeba do PLC nahrát potřebný program a nastavit parametry jednotlivých organizačních bloků.

Pro PLC jsou dostupné dvě verze aplikačního programu. Jedna metoda využívá spojení pomocí T-bloků, což je metoda umožňující navázání TCP spojení, aniž by muselo dojít k restartování PLC a nutnosti přehrání HW konfigurace PLC. Nevýhodou této metody je, že je omezena jen na tyto typy PLC:

- CPU S7 315-2 PN/DP
- CPU S7 317-2 PN/DP
- CPU S7 319-2 PN/DP
- CPU S7 414-3 PN/DP
- CPU S7 416-3 PN/DP
- CPU S7 416-3F PN/DP
- Moduly CP výrobního čísla 6GK7 443-1EX40-0EX0 nebo 6GK7 443-1EX41-0EX0

Druhý typ metody vytvoření komunikačního spojení je určen pro ostatní typy, které tuto první popisovanou metodu spojení neumožňují. Proto je využito nastavení komunikace pomocí konfigurátoru sítě NetPro. Nevýhoda tohoto způsobu spočívá v nutnosti přehrání HW konfigurace a resetu PLC.

Obě popisované komunikační metody fungují na následujícím principu: Pomocí ethernetového spojení „TCP Connection“ se naváže spojení mezi PLC a PC. Poté se z aplikace PC vyšlou do PLC informace o požadovaných signálech, které má PLC při měření poskytovat. Po spuštění měření se ukládají požadované datové vzorky jednotlivých signálů do jedné ze dvou vymezených pamětí PLC. Po naplnění jedné paměti se data vyšlou do PC. V průběhu vysílání dat se dále ukládají procesní data do druhé vymezené paměti. Pro ethernetovou komunikaci jsou v programu v PLC Simatic S7-4xx používány funkce „Send FC 50“ a „Receive FC 60“.

Výhodou této komunikační metody tohoto komunikačního řešení je to, že je část pro PC realizovaná v .NET. Proto lze jednoduše komunikační protokol na straně PC jednoduše modifikovat a zakomponovat do realizace datového rozhraní pro trendový prohlížeč, realizovaný pro firmu Ingeteam a.s. Dalším faktem je skutečnost, že není v tomto případě potřeba platit licenci za využívání přístupu k datům jako v případě využití komerčních produktů. [26]

4 Návrh univerzálního datového rozhraní v prostředí .NET

Jak již bylo zmíněno v úvodu, cílem této práce je navrhnout a vytvořit univerzální datové rozhraní v prostředí určeném pro zajištění přenosu dat ze specifikovaných datových zdrojů do aplikací, které jsou vyvíjeny za pomocí platformy .NET. Návrh a implementace tohoto datového rozhraní je řešen za pomocí objektově orientovaného programovacího jazyka C# a IDE programovací aplikace Microsoft Visual Studio 2008. Navrhované rozhraní je především určeno pro navrhovaný trendový prohlížeč dle požadavků firmy Ingeteam a.s., ale díky univerzálnímu specifikovanému předpisu komunikačních funkcí, lze toto rozhraní využít pro jakékoli jiné aplikace v prostředí .NET, které potřebují získat data z různých datových zdrojů.

4.1 Základní pojmy a možnosti platformy .NET

Jak již bylo zmíněno, tak celý návrh datového prostředí je prováděn za pomocí využití platformy .NET. Podrobný popis platformy je obsáhle téma. V následujících řádcích této podkapitoly je uvedena jen základní problematika, která nutná pro základní přehled informací o popisu použité platformy v rámci návrhu a řešení datového rozhraní.

Platforma .NET Framework

Jedná se o počítačovou platformu vyvinutou firmou Microsoft. Jejím základem je běhové prostředí označené „jako modul CLR“ nebo „běhový systém“ .NET. Jedním z důvodu takto specifikované vytvořené platformy je zjednodušení tvorby programových aplikací s důrazem na přehlednost zdrojových kódů. K tomuto zjednodušení a zpřehlednění přispívá také například možnost, rozdělování kódu na knihovny, podpora komunikací, konektivita s databázemi a podobně. [1] [22]

Dynamická knihovna

Plným názvem dynamicky připojitelná knihovna DLL je označení pro soubor funkcí a procedur, v případě v objektovém programování se může jednat navíc o jednotlivé objekty, datové typy a podobně, které mohou být sdíleny více počítačovými programy. Pomocí knihovny lze usnadnit programátorovi práci při tvorbě programu aplikace pomocí použití již vytvořeného kódu v knihovně, který obsahuje například jednotlivé třídy a metody. Právě problematika této diplomové práce, která se zabývá návrhem a realizací datového rozhraní je řešena za pomocí DLL. [1]

Dědičnost

Pojem, který je základem OOP, neboli objektově orientovaném programování souvisící s třídami a objekty. Z předpisu tříd se vytváří objekty. V obecném příkladu lze tuto dědičnost přirovnat k lidské dědičnosti, ve které děti dědí některé vlastnosti a rysy po svých rodičích. Podobným způsobem to funguje i v OOP programování. Funkce je pak taková, že jednotlivé objekty dědí některé vlastnosti z jiných předešlých objektů. [1]

Třídy, instance, metody a události

Třída je ve své podstatě implementovaný kód, který slouží jako šablona obsahující předpis vlastností a metod. Třída pak slouží pro tvorbu instancí neboli konkrétních objektů, které jsou

v paměti odvozeny z některého vzoru tříd. V objektu je uložen jeho stav formou dat nazývanými atributy. Pro instance tříd - objekty platí, že bývají obvykle vytvářeny pomocí části kódu, nazývajícího se „konstruktor“, který v případě jazyka C# obsahuje klíčové slovo „new“.

Dále objekt musí obsahovat definici komunikačního rozhraní, které slouží pro možnost využití zdrojového kódu objektu. Toto rozhraní lze nazývat pojmem „metody“ objektu. Metody obsahují vstupní parametry neboli argumenty a případný výsledek navracejí pomocí návratové hodnoty. Argumenty se využívají pro zpracování požadavků v těle kódu metody. Návratová hodnota metody je výsledek, který byl vykonán implementovaným kódem v metodě. Při použití objektu není důležitý způsob, kterým objekt požadavky vykonává, ale jakých se pomocí objektu dosáhne výsledků. Tento princip je nazýván zapouzdření objektu.

Jiným typem komunikačního rozhraní objektu je událost, která v objektu slouží pro předání informací části programu, využívající objekt o výskytu nového stavu v objektu. Pro názornost využitelnosti objektu je uveden tento příklad: Používány objekt v trendovém prohlížeči sloužící pro zprostředkování dat vyvolá událost, že jsou připravena nová data. Na tuto událost musí trendový prohlížeč reagovat tak, že nová data vykreslí. [1]

4.2 Shrnutí základních informací pro návrh datového rozhraní za pomocí .NET

Za pomocí prostředí .NET lze podle požadavků vytvořit vlastní vizualizační, archivační, či jiné systémy, které potřebují pracovat se zdrojovými daty z offline nebo online datových zdrojů. V takových případech je vždy potřeba implementovat komunikační metody pro přístupy k těmto zdrojovým datům. Je zde možnost tyto metody implementovat od začátku, nebo pro tento účel využít již implementované komunikační .NET rozhraní, které je implementované například do DLL knihovny. V takovém případě pak koncová aplikace, například trendový prohlížeč získává zdrojová data za pomocí přesně definovaných komunikačních metod použitého datového komunikačního rozhraní. Tímto lze zajistit, že při vytváření jednotlivých aplikací se nemusí v koncové aplikaci specifikovat a programovat metodika čtení k jednotlivým datovým zdrojům. Pro účel získávání dat se pak využívá pouze dostupných komunikačních prostředků. Lze využít například některé komerčně dostupné produkty, které umožňují zprostředkovávat data přímo pro programy vytvořené v pod platformou .NET.

Cílem této práce je navrhnout univerzální datového rozhraní pro poskytování dat trendovému prohlížeči. Rozhraní musí být navrženo koncepčně tak, aby umožňovalo zprostředkovávat data z online i offline datových zdrojů, s možností dodatečného implementování kódu programu pro přístup k různým typům datových zdrojů. Datové rozhraní musí být navrženo a realizováno modulárně.

4.3 Specifikace požadavků pro navrhované datové rozhraní

Návrh datového rozhraní je založen na vlastní koncepci dle specifikovaných požadavků, které vyplynuly při analýze problematiky řešení zpracování dat pro datový prohlížeč. Před začátkem řešení návrhu a realizací datového rozhraní bylo nutné provést konzultaci o požadavcích s firmou Ingeteam a.s.

Pro rozhraní byly specifikovány tyto základní požadavky:

- Vytvoření univerzálního datového rozhraní pro datový prohlížeč historických a reálných trendů tak, aby bylo možno toto rozhraní jednoduše implementovat do aplikací vyvíjených za pomocí prostředků .NET.
- Bylo stanoveno, že tak jako datový prohlížeč bude i datové rozhraní vytvářeno za pomocí jazyka C#.
- Datové rozhraní musí umožňovat zprostředkování z offline a online datových zdrojů.
- Uživatelská konfigurační rozhraní budou realizována v anglickém jazyce.

Na základě těchto základních požadavků bylo potřeba navrhnout koncepční řešení datového prohlížeče. Při řešení a studii problematiky týkající se návrhu a vytvoření kvalitního datového zdroje vznikaly další specifické požadavky, které bylo potřeba do návrhu datového prostředí zahrnout.

Možnost rozšiřitelnosti načítání dat z dalších různých datových zdrojů

Je požadováno, aby koncepce návrhu a realizace datového rozhraní umožňovala dodatečnou implementaci přístupu k dalším, nyní nespecifikovaným datovým zdrojům.

Přístup ke všem typům datových zdrojů pomocí stejných komunikačních metod

Tímto požadavkem je myšleno, aby bylo možno přistupovat ke každému datovému zdroji, pro který je datové rozhraní určeno pomocí principiálně stejných komunikačních metod. Koncovou aplikaci trendového prohlížeče nesmí limitovat různé použití funkcí pro přístupy k datům z různých datových zdrojů. Mírně odlišná může být pouze posloupnost využití jednotlivých metod pro přístup k offline, nebo online datovým zdrojům. O aktuálně používaném typu datového zdroje musí být aplikace trendového prohlížeče využívající popisované rozhraní informována již při prvních krocích načtení informací o zvoleném datovém zdroji.

Možnost výběru periody vzorkovacího kroku a výběru oblasti požadovaných dat pomocí start a stop indexu

V datových zdrojích je obvyklé, že jednotlivé signály mohou obsahovat mnoho datových vzorků. Může se jednat až o miliony vzorků na jeden signál. Při vykreslování vzorků dat za pomocí trendového prohlížeče, který je vytvořen za pomocí .NET může při vykreslování velikého počtu dat docházet k nadbytečnému vytížení PC. Pro aplikaci trendového prohlížeče má smysl poskytovat jen takový počet datových bodů, který odpovídá aktuálnímu nastavení rozlišení obrazovky PC. Díky tomuto způsobu lze docílit v celém trendovém prohlížeči rychlejší odezvy aktualizace a překreslování dat. Druhým případem je také fakt, že uživatel trendového prohlížeče může potřebovat vykreslit jen část datového signálu, pro kterou trendový prohlížeč potřebuje jen specifikovanou oblast vzorků.

S těmito požadavky možnosti výběru vzorkovacího kroku, start a stop indexu se musí počítat už při základním návrhu komunikačních metod datového rozhraní.

Přehledné uživatelské rozhraní pro konfiguraci načítání datových zdrojů

Aplikace využívající vytvořené datové rozhraní by neměla obsahovat vlastní návrh uživatelských oken sloužících pro obsluhu a konfiguraci parametrů datového rozhraní. Návrh datového rozhraní musí být navržen a implementován včetně GUI pro parametrizaci všech typů datových zdrojů.

Filtrace datových signálů

V průmyslové automatizaci se obvykle z řídících procesů archivuje veliké množství signálu. Při prohlížení dat však nemusí být potřeba z důvodu přehlednosti prohlížet všechny signály najednou. V některých případech se mohou archivovat až stovky signálů v jednom datovém zdroji, což by mohlo vést při mnoha signálech k zbytečné nepřehlednosti. Datové rozhraní musí obsahovat filtr, který poslouží k výběru pouze aktuálně potřebných signálů určených pro vykreslení dat.

Stanovení užívání datového rozhraní

Je třeba pevně definovat a stanovit využívání datového rozhraní. Pro datové rozhraní musí být přesně definována přesná funkčnost jednotlivých metod a jejich posloupnost využití sloužící při načítání a vykreslování dat v trendovém prohlížeči. Proto je nutno specifikovat popis pro programátory, který popisuje využití jednotlivých metod rozhraní při návrhu aplikací. Dále je nutno vytvořit nápovědu sloužící pro koncové uživatele, která musí specifikovat ovládání a parametrizaci uživatelského okna datového rozhraní.

Ukládaní konfigurací nastavení datového zdroje

Z výše uvedených požadavků vyplývá, že by měl datový prohlížeč disponovat širokou škálou nastavení, které se týkají parametrizace zvolených datových zdrojů. Lze předpokládat, že uživatel datového rozhraní bude potřebovat načítat některé datové zdroje z patřičných datových signálů s definovanou parametrizací opakovaně. Z tohoto důvodu je nutno navrhnout datové rozhraní tak, aby umožňovalo ukládat a načítat konfiguraci pro načítání datových signálů.

Lze přepokládat, že i v případě koncové aplikace trendového prohlížeče využívající datové rozhraní bude pro zobrazování dat z různých datových zdrojů možno provádět parametrizaci konfigurace z možností ukládání a načítání konfigurace. Bylo rozhodnuto, že je vhodné, aby konfigurace datového rozhraní a koncové aplikace trendového prohlížeče byla pro aktuálně otevřené datové zdroje ukládaná zároveň do jednoho konfiguračního souboru. V případě dodržení této specifikace pak bude možno v případě potřeby znovunačtení již uložené konfigurace v jednom kroku. Nebude tak potřeba manuálních uživatelských parametrizačních zásahů, data z datového zdroje budou načítána dle konfigurace datového rozhraní a pak následně optimálně zobrazena v trendovém prohlížeči dle jeho uložené konfigurace pro jednotlivé křivky. Z hlediska datového rozhraní se může jednat například o parametry, které definují aktuální nastavený typ datového zdroje, výběr konkrétních dat, základní parametrizaci signálů pro datový zdroj. V případě datového zdroje CSV se může jednat například o zvolení cesty k datovému souboru nebo výběru oddělovače. Naopak u online datového zdroje se může jednat o nastavení adres PLC či nastavení dat pro čtení. Dále se pak může jednat o konfiguraci datového filtru signálu, do které spadá nastavení výběru požadovaných datových signálu, které se mají dále používat v datovém prohlížeči.

Jak již bylo zmíněno tak, i pro trendový prohlížeč je nutno provádět ukládání konfigurace, která může úzce souviset s parametrizací datového zdroje. Proto tedy nastal požadavek, aby datové rozhraní dokázalo předat trendovému prohlížeči kompletní aktuálně používanou konfiguraci nastavení.

Archivace aktuálně načítaných dat z online datového zdroje

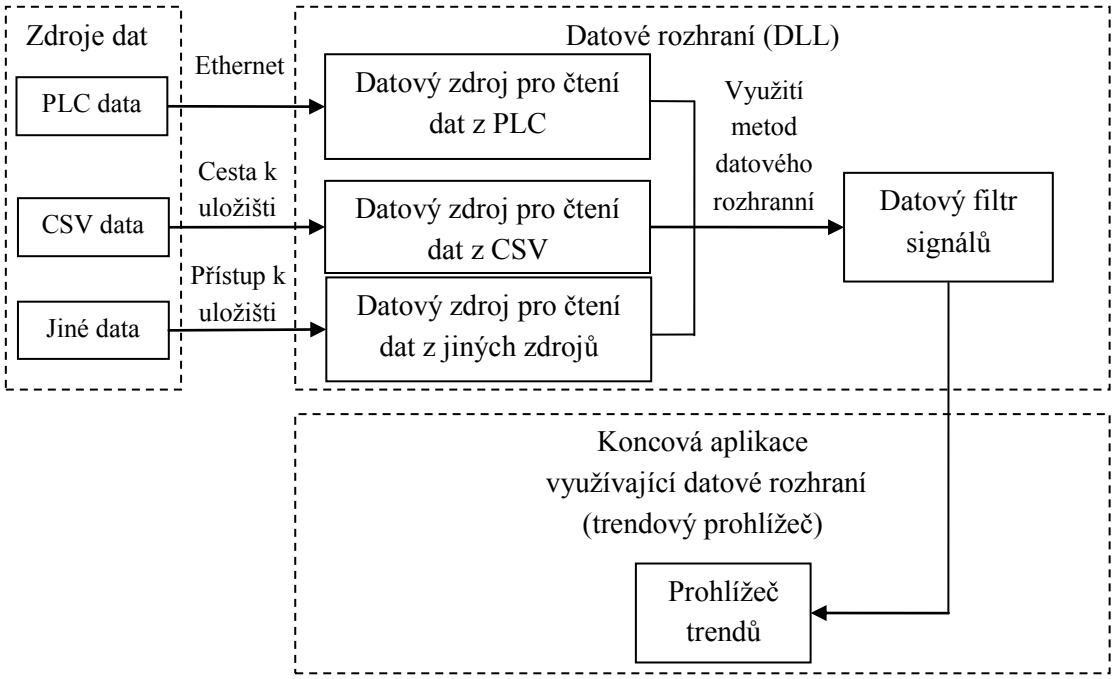
Pod tímto požadavkem je myšleno, aby aplikace datového zdroje umožňovala archivovat načítaná online data, která dále jsou poskytována aplikaci trendového prohlížeče.

4.4 Navržení koncepce dynamické knihovny datového rozhraní

Dle požadavků definovaných v předchozí části bylo nutno specifikovat vlastní návrh a realizaci datového rozhraní pomocí platformy .NET a programovacího jazyka C#. Základním rozhodnutím bylo vytvořit datové rozhraní formou knihovny DLL, což zaručí snadnou implementovatelnost do zdrojového kódu trendového prohlížeče. Veškerá komunikace mezi konečnou aplikací trendového prohlížeče a datovým rozhraním byla navržena univerzálně, tak aby umožňovala doplnění přístupu k různým dalším datovým, za pomocí využití přesně definovaných metod a událostí. Otestování principů navrhovaného rozhraní bylo v navrhovaném datovém rozhraní pro offline zdroj reprezentován přístupem k datovému souboru typu CSV. Otestování připojení k datovému online zdroji bylo navrženo vytvořením připojení k PLC Siemens S7 za pomocí využití firemní komunikační metody [26].

Navržený program datového rozhraní byl dle požadované funkčnosti rozčleněn do několika knihoven. Hlavní knihovna s názvem DataProvider obsahuje interface, neboli předpis metod a událostí sloužících pro stanovení komunikačního prostředku mezi datovým rozhraním a aplikací trendového prohlížeče. Předpis *IDataReader* pak musí využívat všechny ostatní knihovny s třídami datového rozhraní pracující se zdrojovými daty.

V navrhovaném datovém rozhraní se například jedná o třídy, jako jsou *CsvDataReader* nebo *S7OnlineDataReaderTcp* sloužící pro prvotní získání dat z offline nebo online datových zdrojů. Mezi další třídy využívající stanovené předpisy metod dále patří třídy datového filtru *DataFilter*.



Obr. 10 Zjednodušené blokové schéma funkce datového rozhraní

4.5 Rozdělení jednotlivých částí datového rozhraní

Ve specifikaci požadavků bylo rozhodnuto, že rozhraní musí obsahovat pro jednotlivé datové knihovny pracující s daty určitou konfiguraci, čímž je myšlen například výběr datového zdroje, nastavení vzorkovacího kroku signálu, nastavení komunikace pro připojení PLC , nebo nastavení vybraných signálu v datovém filtru. Z tohoto důvodu nastal problém jak vytvořit rozhraní přehledně a univerzálně. Je vhodné, aby každá funkcionalita prohlížeče byla v knihovně zvlášť přehledně oddělena, jak programové části obsahující zpracování zdrojových dat, tak části obsahující GUI sloužící pro konfiguraci jednotlivých parametrů.

Následující text uvádí rozdělení navrhovaných částí datového zdroje do jednotlivých DLL knihoven.

DataProviderConfig

Tato knihovna slouží pro celkovou obsluhu datového rozhraní. Konfigurační rozhraní *UsersControls,* která jsou definována v jednotlivých knihovnách sloužících pro přístup a prací s daty se musí vytvářet ve formuláři *DataProviderConfigForm* podle aktuálního typu výběru datového zdroje. Koncové aplikaci (trendový prohlížeč) je pomocí předání reference na daný objekt pak umožňován přístup k požadovaným datovým zdrojovým datům, dle definovaného komunikačního rozhraní definované v třídě interface *IDataReader*.

Konfigurační požadavky pro GUI v knihovně DataProviderConfig:

- Výběr a zrušení požadovaného typu zdroje.
- Vymezení oblasti pro znázornění vlastní konfigurace pro aktuálně zvolený typ datového zdroje.
- Vymezení oblasti pro konfiguraci datového filtru.
- Zvolení identifikačního jména pro vybraný datový zdroj.
- Celkové potvrzení nebo stornování konfigurace pro datový zdroj.

Informační požadavky pro GUI:

- Přehledné znázornění veškerých informací pro jednotlivé datové zdroje.

DataProvider

Jedná se o knihovnu, která slouží pouze pro uvedení základního předpisu funkcí neboli rozhraní sloužící pro vymezení použití metod pro předávání dat z datových zdrojů. V této části není implementováno žádné GUI. Všechny ostatní níže popisované třídy, které slouží pro zprostředkovávání dat, nebo jakýmkoliv způsobem data modifikují, pak musí ve svém kódu obsahovat metody a události definovaného předpisu funkcí neboli interfacu *IDataReader* implementovaného v této knihovně DataProvider.

CsvDataProvider

Jedná se o knihovnu pro implementaci způsobu načítání dat z datového zdroje CSV, který reprezentuje offline zdroj. CSV soubor je označení pro datový tabulkový formát hodnot, který je oddělovaný specifickými zvolenými znaky. Mezi základní oddělovací znaky patří například , ; : . tabulátor.

Konfigurační požadavky pro GUI v knihovně CsvDataProvider:

- Výběr a načtení požadovaného datového souboru.
- Volba oddělovače signálu v datovém souboru (; , . : tab).
- Výběr desetinné tečky souboru (čárka nebo tečka).

Informační požadavky pro GUI v knihovně CsvDataProvider:

- Zobrazení velikosti načteného souboru.
- Zobrazení informací o načteném souboru.
- Zobrazení počtu signálů a vzorků obsaženém v načteném datovém souboru.
- Zobrazení stavu při načítání a aktualizaci dat.

OnlineProviderS7Tcp

Slouží k načítání dat z online datového zdroje. V tomto případě se jedná o zprostředkování dat z PLC Siemens S7. Pro komunikaci bylo využito fyzického rozhraní Ethernet.

Konfigurační požadavky pro GUI v knihovně OnlineProviderS7Tcp:

- Nastavení IP adresy.
- Nastavení komunikačního portu.

- Volba způsobu zasílání dat.
- Nastavení požadovaných signálů určených pro načítání z PLC.

Informační požadavky pro GUI v knihovně OnlineProviderS7Tcp:

- Informace o stavu navázání spojení s PLC

DataFilter

Tato knihovna je vymezena pro vytvoření třídy pro filtr signálu. Pro přehledný výběr použitých signálů musí GUI v této části využívat tabulkový přehled s jednoduchou obsluhou výběru požadovaných signálů.

Informační požadavky pro GUI v knihovně DataFilter:

- Tabulkový přehled aktuálně načtených signálů.

Konfigurační požadavky pro GUI v knihovně DataFilter:

- Výběr volby využití požadovaných signálů.

4.6 Návrh komunikačního rozhraní v knihovně DataProviderConfig

Hlavní část rozhraní slouží pro celkovou organizaci jednotlivých objektů, které jsou vytvářeny z definovaných tříd datových zdrojů. Návrh rozhraní je proveden tak, že je pomocí této části možno uživatelem nebo načtením známé konfigurace provést nastavení celého datového prohlížeče. Jelikož je tato část zastřešením celého datového prohlížeče, tak by měla být právě zde implementována uživatelská nápověda popisující obsluhu navrženého datového rozhraní. Vyřešení tohoto problému je možno snadno vyřešit pomocí realizací nápovědy formou http stránky, kterou lze na požadavek uživatele stiskem klávesy „F1“ zobrazit ve webovém prohlížeči PC.

V kapitole popisující základní požadavky pro rozhraní bylo zmíněno o požadavku archivace konfigurací trendového prohlížeče a datového rozhraní do jednoho datového souboru XML. Tento problém byl vyřešen za pomocí využití metod zprostředkovávající přenos konfigurací formou objektu. Navržená myšlenka ukládání konfigurace je následující: Trendový prohlížeč získá pomocí předání reference na objekt konfiguraci datového zdroje, doplní svou aktuální konfiguraci pro aktuálně načtený datový zdroj. Následně pak bude proveden zápis této konfigurace sdružené konfigurace do souboru. Při načtení konfigurace pak trendový prohlížeč pomocí níže popisovaných komunikačních metod předá požadovanou konfiguraci datovému rozhraní.

V knihovně DataProviderConfig bylo vytvořeno GUI okno formuláře DataProviderConfigForm, které obsahuje následující komunikační interface.

Komunikační metody pro formulář *DataProviderConfigForm*

Pro možnosti ukládání a načítání dat byly navrhnuty tyto metody.

Metoda pro načtení aktuální konfigurace datového rozhraní do trendového prohlížeče:

Název metody:	*ReadReaderConfig*
Argumenty:	bez argumentu
Návratová hodnota:	objekt s konfigurací datového zdroje

Metoda pro zápis konfigurace z trendového prohlížeče do datového rozhraní:

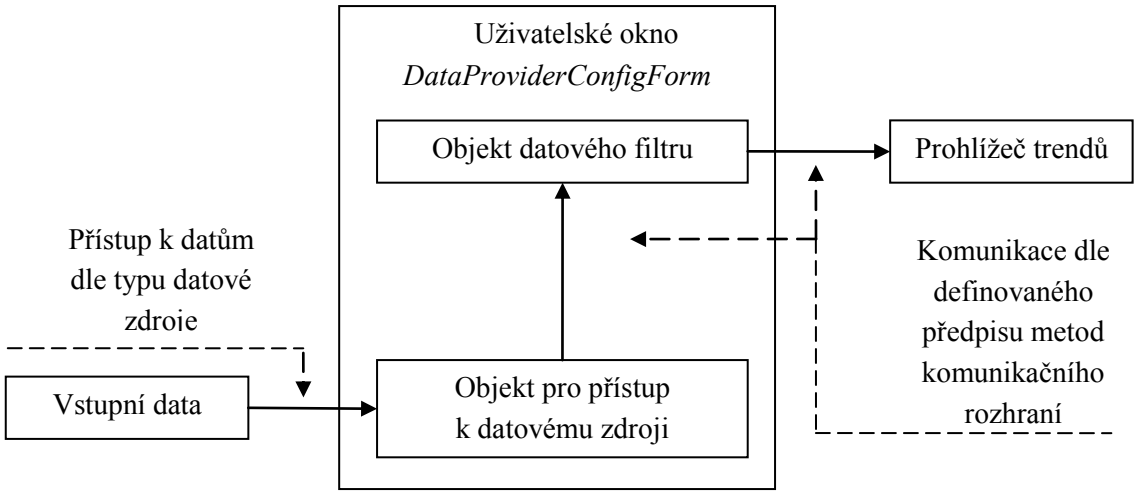
Název metody:	*SetReaderConfig*
Argumenty:	objekt s konfigurací datového zdroje
Návratová hodnota:	bez návratové hodnoty

Komunikační událost

Události slouží pro předání informace o tom, že nastala určitá situace, na kterou je potřeba zareagovat. Trendový prohlížeč je potřeba informovat, když je datové rozhraní připraveno poskytovat data. Pro tento účel byla zvolena metoda vyvolání událostí *DataReady*. Když trendový prohlížeč tuto událost zaznamená, tak je informován o možnosti přistoupení k datům pomocí definovaných komunikačních metod.

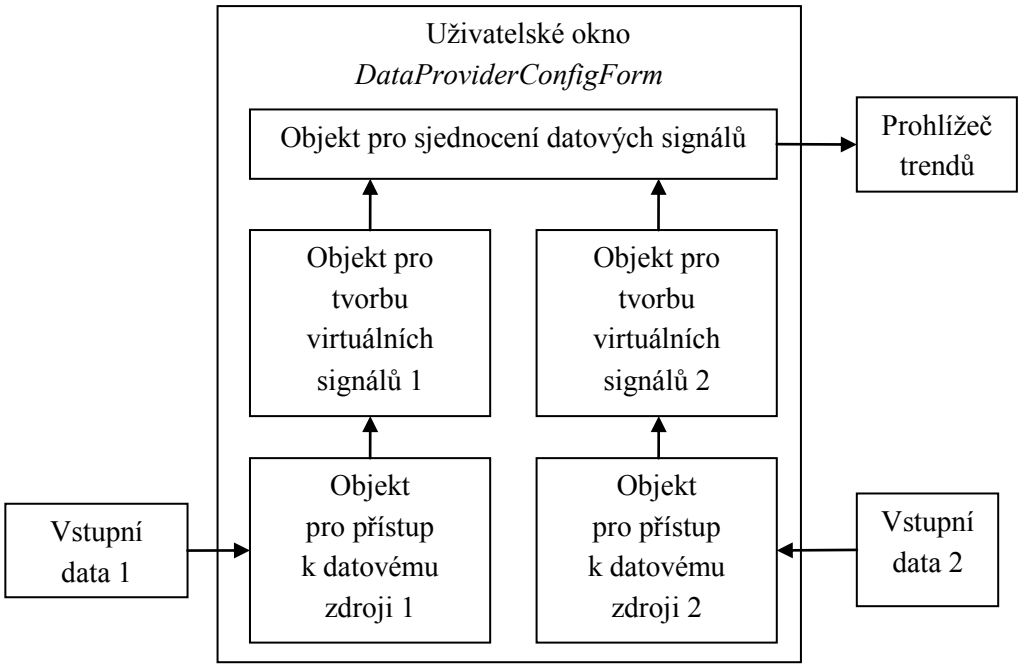
Komunikační referenční objekt

V hlavím části rozhraní se musí vytvářet objekty pro přístup k datům a datového filtru. Každý z těchto objektů datového rozhraní provádí určitým způsobem modifikaci zdrojových dat. Z tohoto důvodu je potřeba předat trendovému prohlížeči informaci o vytvořeném objektu, který provádí v okně formuláře *DataProviderConfigForm* poslední kroky při modifikaci zdrojových dat. Tento problém je vyřešen návrhem veřejného objektu typu *DataInterface* poskytující trendovému prohlížeči informaci o posledním objektu, který je v okně formuláře *DataProviderConfigForm* provádí kroky při modifikaci požadovaných zdrojových dat.



Obr. 11 Obsah hlavní konfigurační části datového rozhraní

Na předchozím Obr. 11je znázorněn návrh datového rozhraní určeného pro implementaci. Jelikož byl návrh datového rozhraní rozčleněn do jednotlivých knihoven a metod, tak lze uvažovat a rozšíření o další vnořené objekty, které mohou dále data upravovat a modifikovat. Takto vytvořené datové rozhraní by mohlo vypadat například tak, jak je znázorněno na Obr. 12.



Obr. 12 Ukázka možnosti jaké rozšiřující možnosti nabízí volba modulárnosti datového zdroje

Pozn. V celé práci je pojednáváno jen o prvním konceptu řešení datového zdroje viz Obr. 11 Důležité bylo navrhnout a otestovat základní koncept návrhu datového rozhraní a otestování metodiky přístupu k online a offline datovému zdroji.

4.7 Návrh předpisu metod komunikačního rozhraní v knihovně DataProvider

Po základních specifikacích struktury datového prohlížeče bylo potřeba začít specifikovat vlastní metody datového rozhraní sloužící k poskytování dat z požadovaných datových zdrojů trendovému prohlížeči. Tento předpis komunikačních metod datového rozhraní byl navržen pro implementaci do knihovny DataProvider.

Pro datové rozhraní pomocí prostředí .NET, byly při návrhu využity možnosti metod a událostí. Pro načítání dat za pomocí datového rozhraní byla definována přesná posloupnost využívání těchto metod a událostí.

Komunikační metody pro definovaný interface *IDataReader*

Na úvod do konkrétní problematiky, jednoduchý příklad využití metody pro navrhované datové rozhraní může vypadat takto:

Trendový prohlížeč potřebuje zjistit počet vzorků obsažených ve vybraném datovém signálu. Pro tento účel je potřeba, aby metoda obsahovala argument udávající číslo signálu. Podle čísla

signálu se v těle kódu funkce vypočte počet vzorků zvoleného signálu. Tato vypočítaná hodnota se následně formou návratové hodnoty metody předá trendovému prohlížeči.

Pro dodržení požadavku univerzálnosti datového rozhraní byl navrhnut pro prácí s daty komunikační interface, který je složen z následujících metod.

Metoda pro zjištění názvů a počtu signálů obsažených v datovém zdroji:

Název metody:	*FindSignals*
Argumenty:	pozice řádku s názvy signálů v datovém zdroji
Návratová hodnota:	textové pole obsahující názvy jednotlivých signálů

Metoda pro zjištění počtu vzorků signálů připravených k načítání do trendového prohlížeče:

Název metody:	*CountValue*
Argument:	pozice zvoleného signálu
Návratová hodnota*:*	počet vzorků

Metoda pro čtení hodnot daného signálu sloužící k předání vlastních hodnot zvoleného signálu:

Název metody:	*GetData*
Argumenty:	ukazatel na pozici v poli signálů
	vzorkovací krok
	počáteční index požadovaných dat
	koncový index požadovaných dat
	formát času - parametr z části datový filtr sloužící pro správné načítání časového formátu
Návratová hodnota:	pole hodnot typu double

Metoda pro získání informací o datovém zdroji:

Název metody:	*GetDescription*
Argumenty:	bez argumentu
Návratová hodnota:	informace o zvoleném datovém zdroji

Metoda pro načtení předchozího datového souboru:

Název metody:	*PreviousFile*
Argumenty:	bez argumentu
Návratová hodnota:	informace zda se jedná o první dostupný datový zdroj

Metoda pro načtení dalšího datového souboru:

Název metody:	*NextFile*
Argumenty:	bez argumentu
Návratová hodnota*:*	informace zda se jedná o poslední dostupný koncový zdroj

Metoda pro zjištění, zda se jedná o offline nebo online datový zdroj.

Název metody:	*CheckOnline*
Argumenty:	bez argumentu
Návratová hodnota*:*	stav datového zdroje

Metoda pro obsluhu online zdroje - spuštění načítání dat z PLC.

Název metody:	*OnlineDataStart*
Argumenty:	bez argumentu
Návratová hodnota:	stav datového zdroje

Metoda pro obsluhu online zdroje - zastavení načítání dat z PLC.

Název metody:	*OnlineDataStop*
Argumenty:	bez argumentu
Návratová hodnota:	stav datového zdroje

Metoda pro aktualizaci dat z datového souboru (Refresh dat pro offline režim).

Název metody:	*Refresh*
Argumenty:	informace zda se má provést přepis konfiguračních parametrů datového zdroje
Návratová hodnota:	stav datového zdroje

Metoda sloužící pro vytvoření signálů s jinou periodou vzorkování.

Název metody:	*MakeDataSample*
Argumenty:	pole řetězců obsahující formát času pro jednotlivé signály
Návratová hodnota:	stav vykonání akce

Tato metoda byla doimplementována v průběhu vývoje datového rozhraní. Metoda slouží k přípravě datových signálů o požadované periodě vzorkovacího kroku dle žádosti trendového prohlížeče. Pro předání požadovaného časového formátu z objektu datového filtru do objektu datového zdroje je v metodě použit argument typu řetězec.

Komunikační události pro definovaný interface *IDataReader*

Vytvořený komunikační předpis datového rozhraní obsahuje dvě události. První událost *DataRefresh* informuje trendový prohlížeč o stavu, že je datové rozhraní připraveno poskytnout nová data. Po příchodu této události může datový prohlížeč začít používat výše popisované metody sloužící pro přístup k datům z dostupných zdrojů. Druhá událost *DataUpdate* slouží pro informování trendového prohlížeče o tom, že datové rozhraní načetlo nová aktualizovaná data z online datového zdroje. Rozdíl při výskytu události *DataUpdate* oproti *DataRefresh* je takový, že trendový prohlížeč nenačítá nově seznam signálů, ale pouze aktualizuje datové vzorky jednotlivých signálů.

4.8 Návrh posloupnosti využití komunikačních metod knihovny DataProvider

Tato podkapitola pojednává o posloupnosti použití definovaných metod pro načtení dat z datového zdroje pomocí definovaného interface *IDataReader*. Navržené datové rozhraní předepisuje dvě posloupnosti. Jedná se o načítání dat z offline a online datového zdroje.

Navržená posloupnost volání metod pro správnou funkčnost načítání dat do trendového prohlížeče pomocí navrženého datového rozhraní - Offline režim

Níže uvedené metody v popisovaném pořadí může trendový prohlížeč pro přístup k požadovaným datům využívat až po příchodu události *DataReady* z hlavní části datového rozhraní - okna formuláře *DataProviderConfigForm.*

1. *CheckOnline* - zjištění zda se jedná o offline/online datový zdroj
2. *GetDescription* - získání informací o datovém zdroji
3. *FindSignals* - zjištění datového pole obsahující názvy jednotlivých signálů obsažených v načteném datovém zdroji
4. *CountValue* - zjištění počtu vzorků, které jsou pro jednotlivé datové signály připraveny
5. *GetData* - načtení pole datových hodnot požadovaného signálu

Dále pak může v tomto režimu aplikace využívající datové rozhraní využívat ostatní funkce

1. *NextFile* - načtení dalšího datového souboru
2. *PreviousFile* - načtení předchozího datového souboru
3. *Refresh* - aktualizace načtených dat pro offline režim

Po zavolání některé z těchto funkcí se v datovém zdroji provede aktualizace potřebných dat. Jakmile jsou nová data připravena, datové rozhraní provede vyvolání události *DataRefresch*. Tímto datový prohlížeč dostane informaci o připravenosti nových dat. Pro obnovení dat je nutno provést v trendovém prohlížeči použití výše popisovaných metod v popisované posloupnosti 2 až 5. Poznámka:

Metodu *MakeDataSample* není potřeba v trendovém prohlížeči využívat, je využívaná jen v rámci jednotlivých částí datového rozhraní.

Navržená posloupnost volání metod pro správnou funkčnost načítání dat do trendového prohlížeče pomocí navrženého datového rozhraní - online režim

Níže uvedené metody v popisovaném pořadí může trendový prohlížeč pro přístup k požadovaným datům využívat až po příchodu události *DataReady* z hlavní části datového rozhraní - okna formuláře *DataProviderConfigForm.*

1. *CheckOnline* - zjištění zda se jedná o offline/online datový zdroj
2. *GetDescription* - získání informací o datovém zdroji
3. *FindSignals* - zjištění datového pole obsahující názvy jednotlivých signálů obsažených v načteném datovém zdroji

Použití dalších metod je možno provádět až po příchodu události *DataUpdate*. Pomocí této události získá trendový prohlížeč informaci o tom, že jsou v datovém rozhraní připravena data z PLC. Poté je nutno v trendovém prohlížeči pro získání dat použít následující metody.

1. *CountValue* - zjištění počtu vzorků obsažených v datovém signálu
2. *GetData* - načtení pole datových hodnot požadovaného signálu

Dále pak může aplikace trendového prohlížeče v tomto režimu využívat ostatní metody datového rozhraní. Jedná se o obsluhu spouštění načítání online dat. Toto lze ovládat pomocí GUI pro online datový zdroj nebo pomocí trendového prohlížeče za pomocí využití definovaných metod datového rozhraní.

1. *OnlineDataStart* - spuštění načítání dat z PLC.
2. *OnlineDataStop* - zastavení načítání dat z PLC.

4.9 Shrnutí funkčnosti a správného užití navrhovaného datového rozhraní

Tato podkapitola pojednává o postupu použití navrhovaného datového rozhraní při využití v realizaci trendového prohlížeče.

Jak již bylo zmíněno v úvodních částech, celý návrh datového rozhraní je rozvržen do DLL knihoven. Pro využití datového rozhraní je potřeba zahrnout tyto knihovny do projektu trendového prohlížeče. Tyto DLL knihovny pak poskytují možnost vytvářet z jednotlivých tříd potřebné objekty, které sloužící pro poskytování požadovaných dat z definovaných datových zdrojů.

Pro načítání dat za pomocí navrhovaného datového rozhraní musí program trendového prohlížeče postupovat v následujících krocích:

1. V prohlížeči se musí vytvořit objekt formuláře *DataProviderConfigForm.*
2. Poté je potřeba v trendovém prohlížeči nutno použít metodu pro zobrazení konfiguračního okna datového rozhraní *DataProviderConfigForm*.
3. Následně je potřeba pomocí konfiguračního okna provést uživatelem konfiguraci parametrů datového zdroje. Provedenou konfiguraci je nutno potvrdit patřičným uživatelským tlačítkem, což zapříčiní vyvolání události *DataReady*. Tímto je trendový prohlížeč informován o připravenosti datového rozhraní.

   V případě, že trendový prohlížeč může poskytnout předchozí uloženou známou konfiguraci datového rozhraní, tak se pomocí patřičné metody provede zápis této konfigurace do vytvořeného objektu datového rozhraní. Poté se v objektu rozhraní provede dle známé konfigurace vytvoření dalších typů objektů. Jedná se o objekty, které jsou vytvářeny z jednotlivých tříd obsažených v knihovnách CsvDataProvider, OnlineProviderS7Tcp, DataFilter.

   V případě načtené konfigurace pro offline datový zdroj dojde podle této konfigurace vytvoření požadovaných objektů a k načtení dat z požadovaného datového zdroje. Následuje automatické vyvolání událostí *DataReady*. Tímto je vyřešen problém, že uživatel nemusí v případě zpětného načítání známé konfigurace do nastavení zasahovat.

Po výskytu události *DataReady* může trendový prohlížeč automatiky začít využívat pro vykreslení křivek dle stanovené posloupnosti definovaný interface *IDataReader*.

V případě načtení konfigurace pro online režim je situace hlavního procesu načítání dat mírně odlišná. Po použití patřičné metody pro předání konfiguračních parametrů dojde v okně uživatelského rozhraní automaticky k vyplnění parametrů, jako například: IP adresa, adresy signálů PLC atd. Uživatel však musí manuálně provést spojení a kontrolu komunikace PLC. Tuto konfiguraci je pak nutno tak jako v plně manuálním režimu parametrizace potvrdit daným uživatelským tlačítkem. Tímto pak dojde k vyvolání události *DataReady*. Dále trendový prohlížeč dle stanovené posloupnosti využije definovaný interface *IDataReader*.

Potřebuje-li trendový prohlížeč za účelem uložení konfigurace do souboru zjistit aktuální stav konfigurace datového rozhraní, použije k tomuto účelu definovanou metodu obsaženou ve vytvořené základní části datového rozhraní.

5 *Realizace datového rozhraní v prostředí .NET*

Tato kapitola pojednává o vlastní implementaci datového rozhraní za pomocí Microsoft Visual Studia 2008 s využitím .NET Framefork verze 3.5. Implementace jednotlivých částí byla provedena přesně tak, jak bylo definováno v předchozí kapitole pojednávající o návrhu datového rozhraní. Pro řešení problémů s implementací byla využita literatura Microsoft Visual C# 2008 [2] a C# 2005:Programujeme profesionálně [1].

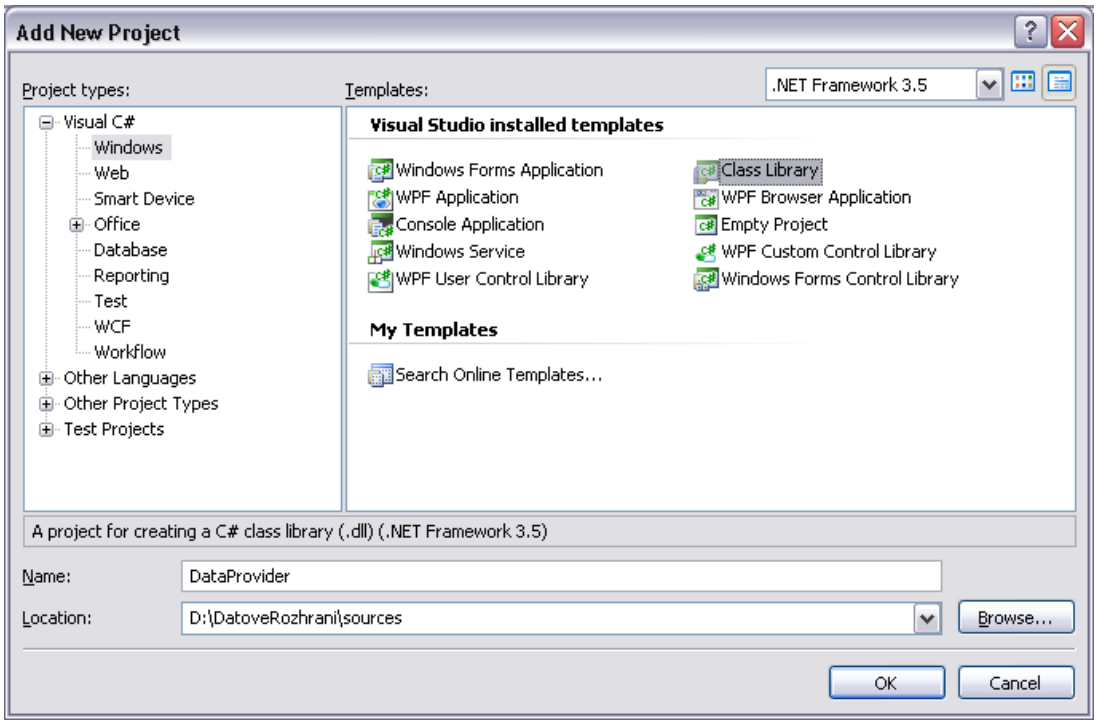
Realizace programu datového rozhraní byla rozčleněna do jednotlivých tříd, které byly implementovány dle uváženého rozvržení do jednotlivých knihoven. Realizace přístupu k offline datovému zdroji datového rozhraní bylo implementováno a testováno za pomocí přístupu k datům s CSV souboru. Realizace pro přístup k datovému zdroji online byla realizována za pomocí TCP/IP spojení s PLC siemens S7.

5.1 Implementace předpisu komunikačních metod pro knihovny rozhraní

V prvním kroku implementace datového rozhraní v jazyku C# bylo potřena vytvořit pro práci s daty předpis komunikačních metod. K tomuto účelu byl vytvořen interface obsahující zmiňovaný předpis metod. Ve vytvořené knihovně DataProvider byl vytvořen podle specifikovaného návrhu interface *IDataReader,* který pak musí být obsažen ve všech částech datového prohlížeče, které umožňují přístup k datovému zdroji a práci s daty.

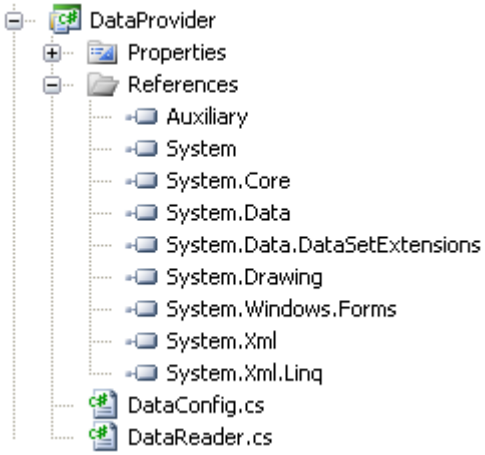
Postup při vytvoření DLL knihovny

Zde je předvedeno vytvoření DLL knihovny za pomocí využití jazyka C# a Microsoft Visual Studia 2008 s využitím .NET network verze 3.5. Konkrétní příklad pojednává o vytvoření knihovny DataProvider, ve které je vytvořen základní komunikační interface *IDataReader*. Princip vytváření ostatních knihoven pro tvorbu datového rozhraní je totožný.



Obr. 13 Hlavní okno pro vytvoření knihovny

Po založení knihovny DataProvider byly vytvořeny jednotlivé třídy. V tomto případě se jedná o třídu *DataConfig*, která obsahuje část kódu potřebného pro ukládání konfigurace datového rozhraní. Druhá třída v knihovně *DataReader* obsahuje již zmiňovaný interface neboli předpis komunikačních metod.



Obr. 14 Sestava třídy „DataProvider“

Základní kód definované třídy

V každé třídě se vyskytuje několik základních částí kódu. Na začátcích každého kódu jsou definovány pomocí klíčového slova *using* využití jmenných prostorů, ve kterých jsou obsaženy potřebné třídy využívané v navrhované třídě. Tyto jmenné prostory musí být v projektu dostupné pomocí přidáním odkazu na patřičnou knihovnu ve složce projetu *References*.

Další částí je právě definice již zmiňovaného jmenného prostoru pomocí klíčového slova *namespace*. Všechny ostatní knihovny realizovaného datového rozhraní jsou pojmenovány totožným názvem jmenného prostoru *DataProvider*. Díky tomuto sjednocení pojmenování jmenných prostorů není potřeba v případě vzájemného využívání knihoven definovat jmenné prostory, ale pouze postačuje přidání odkazu ve složce projektu *References*.

Po těchto částech kódu pak následuje implementace tříd závislých na konkrétním typu knihovny. V této popisované knihovně je implementován předpis komunikačního rozhraní *IDataReader* typu Interface. Tento typ sloužící pouze k definici metod pro využití v jiných třídách datového rozhraní sloužící pro práci s daty. Z tříd tohoto typu nelze vytvářet konečné objekty, ale pouze nové třídy, které tohoto předepsaného rozhraní využívají. V těchto nových třídách je pak ve stejně pojmenovaných metodách jako v předepsaném rozhraní implementován vlastní kód dle určení jednotlivých tříd.

Ukázka kódu programu definice rozhraní v jazyku C#:

```
using System;
namespace DataProvider
{
    public interface IDataReader
    {
      //Implementace událostí
      event EventHandler DataUpdate;
      event EventHandler DataRefresh;
      //Implemetace metod
      string[] FindSignals(int SigPos);
      int CountValue(int SigPos);
      double[] GetData(int SigPos, int SampleStep,int StartIndex, int
                       EndIndex, string Time);
      string GetDescription();
      bool Previous();
      bool Next();
      bool CheckOnline();
      bool OnlineDataStart();
      bool OnlineDataStop();
      bool Refresh(bool CsvGuiRefresh)
      bool MakeDataSample(string[] TimeType);
    }
}
```

5.2 Implementace hlavní částí datového rozhraní

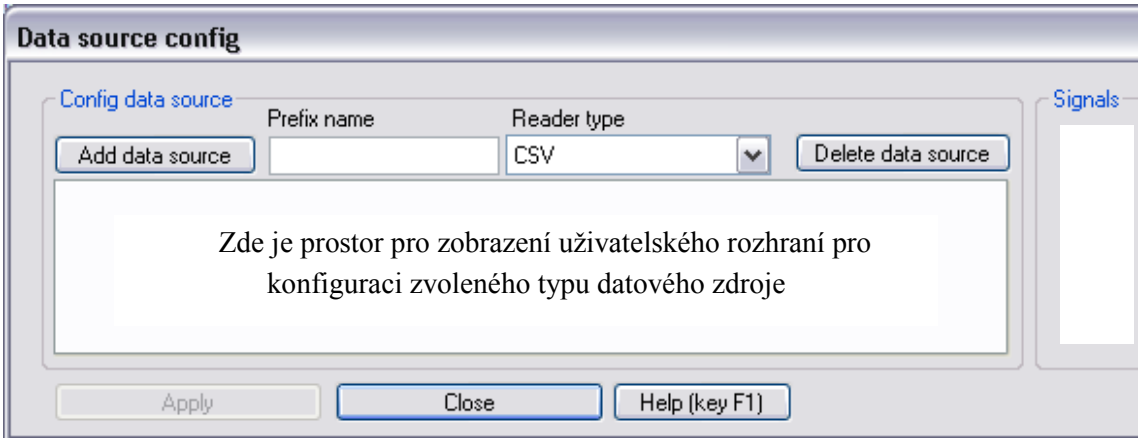
Uživatelské konfigurační rozhraní

V navrhovaném datovém rozhraní bylo potřeba implementovat část obsahující konfiguraci celého datového rozhraní. Tím je myšlen výběr a parametrizace datového zdroje, dále pak nastavení datového filtru. Uživatelská rozhraní, která byla implementována do jednotlivých knihoven datových zdrojů, pak musí být v této hlavní části datového rozhraní dle požadavků konfigurace datového zdroje vyobrazena.

Pro tento účel bylo vytvořeno hlavní konfigurační okno, které slouží pro celkové ovládaní datového zdroje. V tomto okně prohlížeče se tvoří jednotlivé objekty „UsesControl“, sloužící pro konfiguraci jednotlivých typů datových zdrojů.

Výběr datového zdroje je prováděn pomocí komponenty „Combo box“. Dále je pak možno zapsat do komponenty „Text box“ s označením „Prefix name“ jméno, které slouží pro identifikaci přidaného datového zdroje. V aktuální verzi navrhovaného datového rozhraní je najednou možno nakonfigurovat pouze jeden datový zdroj. Koncepce rozhraní je však navrhována tak, aby bylo možno rozhraní případně rozšířit konfigurací z více datových zdrojů, proto je parametr identifikace „Prefix name“ pro datový zdroj důležitý. Po výběru typu datového zdroje a zapsání „Prefix“ jména se stiskem tlačítka „Add data source“ konfigurační rozhraní přidá vybraný datový zdroj do komponenty „Tab Control“.

Obsažená komponenta „Group Box“ slouží pro seskupení jednotlivých komponent dle jejich funkčnosti. V tomto GUI je využita pro „Config data source“ a „Signals“. Tato komponenta se využívá pak i v konfiguracích jednotlivých knihoven datových zdrojů.



Obr. 15 Hlavní konfigurační okno datových zdrojů

Implementace kódu komunikačního rozhraní v definované třídě základního konfiguračního okna.

Ukázka kódu pro definované rozhraní hlavního komunikačního okna:

```
// Definice jmenného prostoru.
namespace Ingeteam.DataProvider
{
   // Vytvoření třídy formuláře.
   public partial class DataProviderConfigForm : Form
   {        // Vytvoření veřejného objektu, pomocí kterého pak datový
            prohlížeč přistupuje k požadovaným datům (v řešené verzi
            prgramu je posledním článkem rozhraní objek typu data
            filter).
            public DataFilter DataReader = null;
            //Metoda sloužící pro nastavení datového rozhraní dle
           předání konfiguračního objektu
            public void ReaderConfig(DataConfig ReaderConfig)
            {
            // Zde je obsažen kód metody provádějící tvorbu potřebných
            objektů a uživatelského rozhraní.
            }
            //Metoda sloužící pro předání aktuálního stavu
            nastavení datového rozhraní
            public DataConfig ReaderConfig()
            {
            // Tato metoda pomocí reference na místní objekt obsahující
            aktuální konfiguraci datového rozhraní.
            return mDataConfig;
            }
    }
}
```

5.3 Rozčlenění jednotlivých částí komunikačních knihoven datového rozhraní

V kapitole týkající se návrhu rozhraní je uvedena potřeba členění jednotlivých částí do knihoven. Proto bylo potřeba blíže specifikovat obsahy těchto knihoven do těchto jednotlivých tříd:

Uživatelské okno z konfigurací datového zdroje

Slouží pro přehledné ovládání, konfiguraci a parametrizaci datového rozhraní. Instance těchto „UsesControl“ jsou pak vytvářeny hlavním konfiguračním v okně formuláře *DataProviderConfigForm.*

Zdrojový kód pro práci s daty

Jedná se o části, které obsahují vlastní implementaci kódu do patřičných tříd pro načítání a poskytování dat pro trendový prohlížeč.

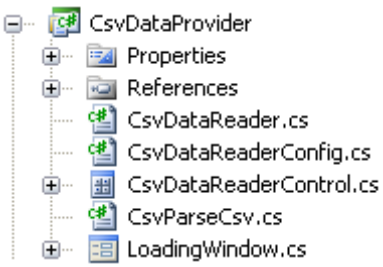
V této části jsou implementovány jak metody předpisu datového rozhraní, tak metody privátní, sloužící v rámci programu datového prohlížeče.

Definice parametrů

Část obsahující třídu s globálními proměnnými, která je nutná z důvodu možnosti přístupu k jednotlivým proměnným z různých částí kódu datového rozhraní. Pomocí těchto globálních poměných je pak možno provádět ukládání a načítání parametrizace jednotlivých částí datového rozhraní. Tyto globální proměnné jsou zapouzdřeny do objektu tak, aby jej bylo možno ukládat neboli „serializovat“ v trendovém prohlížeči pomocí formátu XML.

Ostatní potřebné třídy pro definované v knihovně

Dle typů knihovny pak být mohou jednotlivé části kódu z důvodu přehlednosti implementovány do vlastních tříd.



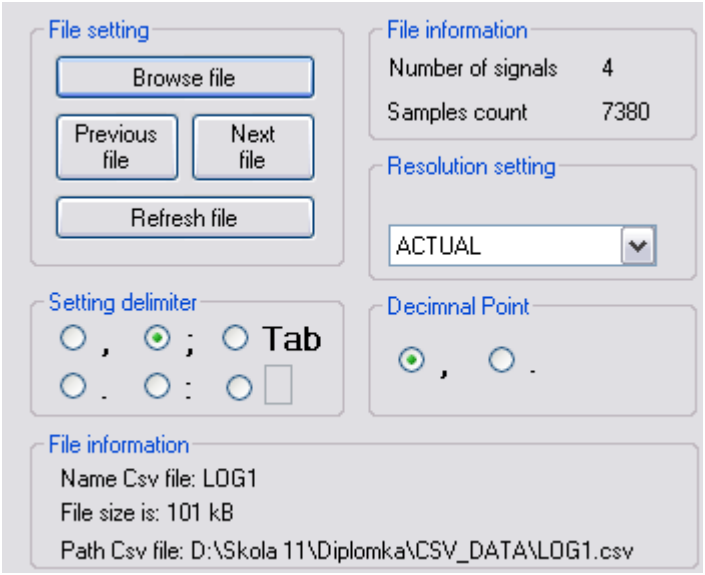
Obr. 16 Příklad sestavy knihovny pro načítání z datového zdroje CSV

5.4 Implementace knihovny poskytující offline data z CSV souboru

Tato podkapitola pojednává o realizaci vytvoření DLL knihovny, která v realizovaném datovém rozhraní slouží k reprezentaci načítání dat pomocí offline datového zdroje CSV.

Uživatelské okno z konfigurací datového zdroje

Pro přehlednost byly tyto požadované konfigurace v GUI rozčleněny za pomocí komponenty „Group Box“ do následujících skupin, které pak obsahují tyto další ovládací komponenty „Button“, „Radio button“, „Label“, „Combo box“.



Obr. 17 Definované „Users Controls“ pro konfiguraci datového zdroje CSV

Po spuštění konfiguračního okna se zobrazí jednotlivé grafické komponenty. Některé komponenty jsou zprvu neaktivní a aktivují se až po načtení signálu.

Použití „Uses Control“ pro konfiguraci datového zdroje CSV

Uživatel musí pro konfiguraci podstoupit následující kroky:

1. Pomocí nastavení parametru „Setting delimiter“ se provede nastavení znaku pro oddělení jednotlivých signálů obsažených v datovém souboru. Dále pak je nutno nastavit parametr „Decimal point“, který slouží pro nastavení oddělení desetinného čísla v načítaných datech.
2. Nastavení parametrů pro způsob vzorkování „Resolution settings“. Důležitost tohoto parametru slouží pro případ, ve kterém je v datovém rozhraní načteno mnoho vzorků signálu. V tomto případě pak může trendový prohlížeč požadovat načítání dat s vyšší periodou vzorkovacího kroku než jedna, což může vést při vykreslování křivek ke vzniku aliasingu. Perioda tohoto kroku závisí na aktuálním nastavení rozlišení obrazovky PC a dostupných načtených datech datového zdroje. Aby nedošlo ke ztrátě datových špiček ležících mimo periodu vzorkovacího kroku, je

potřeba v datovém rozhraní připravit data s požadovanou vzorkovací periodou dle požadovaného nastavení uživatele. V případě nastavení „ACTUAL“ se další data nepřipravují. Uživatel pak musí počítat při vykreslení celého náhledu signálu s možnou chybou datových špiček (viz. Obr. 27, Obr. 30). Výhoda této volby spočívá v rychlosti načítání dat, protože datové rozhraní nemusí procházet všechna data a vytvářet nové datové signály dle zvolené konfigurace. Pro poskytnutí všech špičkových hodnot, nacházejících se mezi požadovanou volbou periody vzorkování, která je volená trendovým prohlížečem je potřeba zvolit jiné nastavení parametru „Resolution settings“. Je pak možno vybrat z následujících možností:

„MIN“ - Poskytnutí minimálních hodnot signálů vzorovací periody

„MAX“ - Poskytnutí maximálních hodnot signálů vzorovací periody

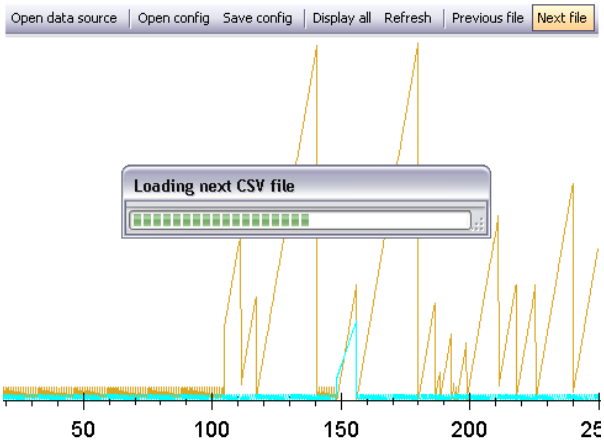
„MIN_MAX“ - Poskytnutí maximálních a minimálních hodnot signálů se zdvojnásobenou vzorovací periodou.

3. Výběr datového zdroje pomocí uživatelského tlačítka „Browse file“.
4. V okně výběru datového souboru se vybere pořadovný soubor.
5. V případě potřeby lze uživatelskými tlačítky „Next file, Previous file“ načítat ostatní datové soubory které jsou obsaženy ve složce aktuálně načteného souboru. Tyto funkce lze provádět z trendového prohlížeče za pomocí využití definovaných metod datového rozhraní.

Okno Loading Window informující o průběhu zpracovávání dat

Pro načítání nebo zpracovávání většího množství dat je vhodné uživatele informovat o tom, že aplikace datového rozhraní není bez činnosti, ale zpracovává požadovaný úkol. K tomuto účelu bylo využito komponenty „Progress bar“. Bylo potřeba zajistit znázornění situace, kdy datové rozhraní zpracovává data i pro případ, ve kterém není hlavní konfigurační okno s konfigurací pro datový zdroj CSV zobrazeno, ale trendovým prohlížečem je tento datový zdroj využíván. Trendový prohlížeč může pomocí definovaného rozhraní zadat požadavek pro aktualizaci dat, nebo načtení dalšího datového zdroje. Datové rozhraní pak začíná zpracovávat požadavek. Nad oknem aplikace trendového prohlížeče zobrazí samostatné okno obsahující komponentu „Progess bar“, ve kterém probíhá změna hodnoty dle aktuálního stavu běhu datového rozhraní.

Na níže uvedeném obrázku je znázorněno vyvolané okno, které signalizuje načítání dat. V uvedeném případě se jedná o okno vyvolané nad oknem trendového prohlížeče při požadavku načtení dalšího datového souboru.



Obr. 18 Zobrazené okno „LoadingWindow“ signalizující zpracovávání dat nad oknem trendového prohlížeče

Definované části pro načítání z offline CSV datového zdroje

Z hlediska přehlednosti byl program pro CSV datový zdroj rozdělen do několika tříd. Hlavní třída datového zdroje *CsvDataReader*. Tato třída využívá komunikační předpis interface *IDataReader* a slouží pro celkovou zprávu načítání dat z datového zdroje CSV. Dále pak třída *ParseCsv*, která slouží k čtení dat z datového souboru CSV. Dále je zde pak implementována třída obsahující globální proměnné *CsvDataReaderConfig*. V poslední řadě se jedná o okna formulářů *CsvDataReaderControl* a *LoadingWindow,* určená pro konfiguraci zdroje a znázornění stavu procesu v datovém rozhraní.

Způsob načítání dat z CSV – použitá třída pro načítání dat do listu řetězců

Pro načtení dat z datového souboru CSV byla definována třída ParseCsv. V této třídě jsou požadovaná data načítána do listu řetězců pomocí vytvořeného objekt typu StreamReader, který je definován v systémové knihovně SystemIO. Tento objekt pro načítání dat byl pak zapouzdřen do vytvořené metody *ParseCsv*. Vstupními argumenty vytvořené metody jsou cesta k datovému souboru CSV uloženému na disku PC, dále pak oddělovací znak sloupců dat (signálů datových zdrojů). Tato metoda je používána v třídě *CsvDataReader*. Vstupními argumenty jsou parametry z GUI pro konfiguraci datového zdroje CSV. Návratová hodnota metody typu bool informuje o správnosti načtení datového zdroje.

Ukázka kódu definované metody ve třídě „ParseCsv“:

```
public bool ParsingCsv(string path, char delimiter)
```

Metoda pak ukládá data z čteného souboru do definovaného listu stringů (řetězců):

```
public List<string[]> ParsedData = new List<string[]>();
```

Ve třídě „CsvDataReader “ je pak vytvořen objekt pro pasování dat „mParseCsv“

```
public CsvParseCsv mParseCsv = new CsvParseCsv();
```

Využití metod a událostí komunikačního předpisu ve třídě datového zdroje *CsvDataReader*

Ukázka kódu definice třídy *CsvDataReader* odvozené z rozhraní „IDataReader“:

```
public class CsvDataReader : IDataReader
{
// Zde je obsažen vlastní kód třídy.
}
```

V následující části je pojednáno, jakým způsobem je využit definovaný interface, neboli komunikační předpis *IDataReader* pro metody implementované ve třídě *CsvDateReader*.

V některých případech metody jen poskytují aktuální stav požadované proměnné a v datovém rozhraní neprovádí žádné výrazné změny. Naopak použití některých metod provádí před poskytnutím návratové hodnoty různé procesy s daty nebo datovým rozhraním.

U všech definovaných metod v konkrétní třídě je vždy před každým argumentem znak „a“. Toto označení slouží jednoznačnosti odlišení argumentů od ostatních proměnných obsažených ve třídě. Pro úplnost, proměnné pro využití v celé třídě byly před názvem označovány znakem „m“. Proměnné definované v metodách znakem „l“. Veřejné proměnné označené klíčovým slovem „public“ byly pojmenovávány bez předpony.

Implementace jednotlivých metod komunikačního rozhraní pro datový zdroj CSV je provedena následovně:

event EventHandler DataUpdate

Tato událost je v datovém zdroji *CsvDataReader* využita při načtení nebo aktualizaci dat ze zdrojového datového souboru „CSV“.

event *EventHandler DataRefresh*

Tato událost není pro datový zdroj *CsvDataReader* využita.

bool CheckOnline()

Jelikož se v tomto případě jedná o offline zdroj, tak metoda navrátí hodnotu „false“.

public string[] FindSignals(int aSigPos)

Metoda zjistí z listu *mParseCsv.ParsedData* načtená jména datových signálů. Vstupní argument *aSigPos* slouží k definici řádků, ve kterém jsou obsaženy požadované názvy signálů. V případě potřeby náhledu na formát dat lze navrátit v poli žetězců například 1. řádek datových hodnot (využito u data pro náhled na data v třídě datového filtru).

public int CountValue (int aSigPos)

V metodě se provede jen výpočet indexů vzorků signálů. Návratová metoda informuje trendový prohlížeč o počtu dostupných dat pro vykreslení křivek.

public double[] GetData (int aSigPos, int aSampleStep,int aStartIndex, int aEndIndex, string aTime)

Metoda slouží k navrácení konečného datového pole dat sloužícího pro vykreslení jedné datové křivky v trendovém prohlížeči. Počet opakování využití této metody pro jedno

vykreslení všech dat v trendovém prohlížeči je totožný s počtem obsažených signálů v načteném datovém zdroji.

Datové pole je navráceno s ohledem na paramtery vstupních argumentů metody, které definují nastavení počátečního, koncového indexu a vzokovacího kroku. Bylo stanoveno že, požadovaná hodnota periody vzokovacího kroku bude z trendového prohlížeče pouze z řady číselných hodnot 0,1,2,4,8,16,32….. Důvod tohoto rozhodnutí vyplýva ze způsobu přípravy nových datových signálů eliminující aliasing. Bližší informace jsou uvedeny v popise metody *MakeDataSamples*.

V trendovém prohlížeči lze zobrazit jen takové maximální množství vzorkovacích bodů, které odpovídá rozlišení obrazovky PC. Velikost načtených předaných návratovou hodnotou metody závisí na vztahu počtu vzorků/požadovaná perioda vzorkovacího kroku. Tato perioda závisí na trendovém prohlížeči, který jej musí pro optimalizaci počítat dle aktuálního nastavení „zoomu“ a rozlišení obrazovky PC. Zde je uveden konkrétní příklad:

Z hlediska vzorkování je podstatné pouze horizontální rozlišení. V případě rozlišení obrazovky 1920 horizontální x 1080 vertikální a načteného datového zdroje, z například 5 000 000 vzorků a 20 datových signálů dojde při vykreslování všech vrorků k prodlevě. Pro vykrelení tohoto celého modelového datového signálu s 5 000 000 vzorky pak odpovídá 2604 vzroků na jeden obrazový bod. Z této jednoduché analýzy vyplývá, že je pro náhled celého datového signálu přenášeno veliké množstnví nadbytečných nevyužitých dat, které ve výsledku nelze vykreslit a splynou do jednoho obrazového bodu.

Díky předpokladu, že trendový prohlížeč bude žádat data s patřičnou periodou vzorkovacího kroku, pak nedojde ani v případě načtení obsáhlého datového zdroje s mnoha vzorky k přehlcení nadbytečnými daty, které není možno pří náhledu na všechna data z důvodu splynutí jednotlivých bodů pixelů obrazovky v trendovém prohlížeči zobrazit. Tento způsob načítání je výhodný z hlediska optimalizace rychlosti a využití oblasti paměti PC, avšak je jej třeba používat obezřetně, jelikož věrohodnost dat závisí na tvaru signálů, může totiž dojít k výskytu aliasingu neboli podvzorkoviní signálů. Toto může způsobit nesprávné vykreslení signálů, případně nevykreslení v zobrazeném signálu lokální minima nebo maxima. Ošetření tohoto nepříznivého jevu zkreslení signálu je provedeno kódem programu, který je implementován do zvláštní dodatečné metody definovaného rozhraní s názvem *MakeDataSaples* . V této metodě se pak připraví požadavaná data dle nastavení parametru GUI CSV „Resilition Setting“. V případě nastavení „ACTUAL“ se metoda *MakeDataSaples* nevyužívá a parsování dat se provádí v rámci této popisované metody *GetData*. V případě nastavení „Resilition Setting“ na jednu z hodnot „MIN,MAX, MIN_MAX“ je v první fázi provedena pomocí metody *MakeDataSaples* požadovaná příprava dat z listu dat řetězců *mParseCsv.ParsedData* do listu dvourozměrného pole typu double. V tomto případě je pak v metodě *GetData* navráceno pouze připravené datové pole dle požadovaného vzorkovacího kroku.

Jak již bylo zmíněno, v případě nastavení „Resilition Setting“ na hodnotu „ACTUAL“ jsou dle vstupních parametrických argumentů metody *GetData* data z načteného listu řetězců *mParseCsv.ParsedData* parsována do jednorozměrného datodého pole typu „double“. Aby byla data správně načítána dle správných formatovacích znaků, tak bylo potřeba pro tuto funkčnost vytvořit třídu *ValueParser*. Navržené datové rozhraní poskytuje veškeré datové signály v číselném formátu datového typu „double“. Jestliže data obsahují datový formát typu čas, tak jsou data z datového zdroje parsována do pole typu „double“ podle vstupního vzoru řetězce

z argumentu metody *aTime*. Takto formatovaná data pak trendový prohlížeč umožňuje zpětně znázornit ve formátu času.

public string GetDescription()

Tato medoda navrátí pouze navrátí cestu k aktuálně načtenému datovému souboru.

public bool Refresh(bool aCsvGuiRefresh)

Navržená metoda byla navržena tak, aby ji bylo možno využít, jak pro prvotní načtení dle požadavků datového rozhraní, tak pro aktualizaci dat na základě požadavků z trendového prohlížeče. Při prvotním načítání se nastavení datové konfigurace nastavuje vždy do výchozího stavu, ale při požadavku aktualizaci dat musí zůstat zachována stávahící konfigurace nastavených parametrů. Pro předání této informace o požadavku přestavení konfigurace slouží argumet metody *aCsvGuiRefresh*. Hodnota „true“ - konfigurace do výchozího stavu stavu, „false“ - stávající konfiguraci ponechat.

V případě, že celé načtení dat proběhne v pořádku je návratová hodnota metody true, v případě výskytu chyb false. Metoda sloužící pro prvotní načtení a aktualizaci dat z datového souboru „CSV“ je řešena následujícím způsobem. Načítání dat probíhá za pomocí využití již zmiňované třídy *ParseCsv* a metody *ParsingCsv*.

V případě, že jsou data načtená v pořádku, tak se v posledním kroku metody se provede vyvolání interrface události *DataRefresh*. Na tuto událost zareaguje datový filtr tak, že se aktualizují data v tabulce signálů. V případě, že se jedná o stav, kdy byla metoda vyvolaná z trendového prohlížeče, lze poté provést aktualizaci vykreslení křivek.

public bool Previous(),public bool Next()

Tyto obě metody slouží pro načítání dalšího a předchozího datového souboru CSV. Pro práci s datovým souborem využívá metoda knihovnu „*SystemIO*“ tak jako v případě výše popisované třídy *ParseCsv*. Při využití metody se zjistí všechny dostupné soubory CSV pro aktuálně používanou složku dat. Přístupové cesty ke všem „CSV“ souborům se uloží do pole řetězců.

Provede zjištění, která pozice v poli přístupových cest *lFileName* odpovídá aktuálně používanému datovému souboru. Dále se zjistí, zda se zrovna nejedná o poslední nebo první datový soubor v používaném adresáři. V případě , že zrovna není načten poslední nebo první datový soubor, tak se provede posun na požadovaný indexu v poli řetězců *lFileName*. Po tomto kroku je dostupná nová přístupová cesta k dalšímu, nebo předchozímu datovému souboru. Poté se v metodě použije metoda *Refresch(true)*. Jedná se o načtení úplně nového zdrojového souboru, a proto je potřeba přepsat dosavadní konfiguraci datového zdroje.

Návratová hodnota informuje trendový prohlížeč o stavu, zda je aktuálně načtený datový soubor první nebo poslední z aktuálně používaného aresáře datových souborů. V případě, že se jedná o načtený krajní datový soubor předpokládá se, že trendový prohlížeč zakáže uživateli používat uživatelské tlačítko „Previous“ nebo „Next“. Tyto metody jsou používány pro procházení datových souborů i v GUI pro konfiguraci CSV datového zdroje.

public bool MakeDataSamples(string[] TimeType)

Základní problematika o nutnosti vzorkovacích kroků byla již zmínka při popisu metody *GetData*. Tato metoda vznikla dodatečně na základě zpracování požadavků v případě nastavení

parametru „Resilition Setting“ v GUI pro datový zdroj CSV na jednu z hodnot „MIN,MAX, MIN_MAX“. Poté je nutno poskytnout trendovému prohlížeči data s požadovanou periodou vzorkovací frekvence s důrazem na odstranění jevu aliasingu, který může mít za následek špatné vykreslení lokálnícho minima a maxima mezi periodami vzorkovacího kroku vykreslovaného signálu.

Metodu *GetData* používá trendový prohlížeč při načítání dat pro každý signál jednotlivě. V případě posunu zobrazení trendu dle požadavků uživatele po časové ose nebo zoomování, kdy se mění pro předávaná data start, stop index a perioda vzorkovacího kroku by bylo nutno znova připravovat nová filtrovaná data, docházelo by k dlouhým odezvám. Tento způsob by zcela znehodnotil myšlenku využití volby periody vzorkovacího kroku sloužící pro zrychlení běhu trendového prohlížeče. Aktualizace dat při procházení vykreslených křivek by byla pomalá.

Z tohoto důvodu bylo navrhnuto následující řešení. Pro přípravu dat s potlačeným jevem aliasingu bylo navrhnuto, že všechny načtené datové signály musí být pro trendový prohlížeč připraveny dle očekávaných voleb period vzorkovacích kroků najednou, před prvním poskytnutím dat do trendového prohlížeče. Vstupními daty pro přípravu dat jsou data z listu dat řetězců *mParseCsv.ParsedData*. Při procházení a „parsování“ jednotlivých vzorků je uživateli zobrazeno okno „LoadingWindow“ signalizující běh procesu načítání dat. V metodě je nutno využít pro správnou funkci parsování časových formátů informaci o uživatelky nadefinovaném předpisu datového formátu času z objektu datového filtru. Tento problém byl vyřešen vstupním argumentem metody sloužící pro předání formátu všech signálů pomocí pole stringů.

Poznámka: Metoda *GetData* pracuje vždy jen s jedním signálem – proto je argument jen typu string. Tato metoda *MakeDataSample* pracuje se všemi signály, proto je argument v metodě typu pole stringů.

Kroky při přípravě dat pomocí metody *MakeDataSamples*:

1. Zjistí se aktuální nastavení rozlišení obrazovky PC.
2. Provede se zjištění počtu vzorků pro jednotlivé signály načtené signály v listu *mParseCsv.ParsedData*.
3. Provede se výpočet kolik bude potřeba vytvořit nových pomocných signálů pro jeden skutečný datový signál. Je potřeba vytvořit takový počet pomocných datových signálů, aby rozhraní umožňovalo poskytnout data pro trendový prohlížeč pro všechny očekávané periody vzorkovacího kroku.

   Tento požadavek je v kódu programu proveden následovně:

```
int lSignalFieldCount = 1;
int lSampleCount = mSampleCount;
while ((lSampleCount/2) > Screen.PrimaryScreen.Bounds.Width)
      {
          lSampleCount = lSampleCount / 2;
          lSignalFieldCount++;
      }
```

4. Poté se dle vypočtené hodnoty *lSignalFieldCount* definující počet požadovaných pomocných signálů vytvoří dvourozměrné pole.
5. Jednotlivá pomocná datová pole se naplní požadovanými daty dle nastavení parametrů „Resilition Setting“ v GUI CSV. Prochází se všechny vzorky signálů a vybírají se lokální extrémy mezi vzorkovací periodou.
   Na následujícím obrázku Obr. 19 je znázorněn výběr vzorků s vzorkovací periodou čtyři. V případě požadavku „MIN_MAX“ musí trendový prohlížeč při zvoleném vzorkovacím kroku dodržet předpokládaný počet vzorků, z tohoto důvodu se vzorky musí v datovém rozhraní počítat z dvojnásobné vzorkovací periody.
   Na následujícím obrázku je znázorněn v prvním řádku zdrojový signál.
   Na druhém řádku je znázorněn výběr minim pro pomocný generovaný signál se zvolenou hodnotou periody vzorkovacího kroku čtyři. Třetí řádek pak znázorňuje výběr maxim, a čtvrtý řádek výběr minim a maxim z dvojnásobného vzorkovacího kroku.

Obr. 19 Příklad znázornění výběru vzorků pro signál s vzorkovacím krokem 4

Jelikož je potřeba tento popisovaný proces aplikovat na všechny načtené datové signály z načteného zdrojového datového souboru, proto jsou jednotlivé generované signály uloženy do listu dvourozměrného datového pole.

Kód programu pro vytvoření listu dvourozměrných datových polí:

```
List<double[][]> lAllSignals = new List<double[][]>();
```

Poté jsou data, která neobsahují chyby aliasingu připravena pro předaní trendovému prohlížeči.

Praktický příklad:

Načtený signál obsahuje 20 000 vzorků, horizontální rozlišení obrazovky PC je 1920 pixelů. Pro tento případ budou vytvořeny celkem čtyři pomocné signály s vzorkovací periodou 1,2,4,8

Obr. 20 Grafické znázornění velikosti požadovaných pomocných polí

Na Obr. 20 je uvedeno grafické znázornění velikosti požadovaných pomocných polí, se znázorněním potřeby rezervace, přibližně dvakrát většího paměťového prostor pro každý datový signál (data vzorkovacího kroku 1 + ostatní vzorkovací kroky).

Jako nejmenší počet vzoků bylo v přikladu stanoveno 2500, prototože 2500/ 2 = 1250 což to je už pod hranicí rozlišení v uvedeném příkladu 1920 pixelů. Bylo rozhodnuto, že datové rozhraní bude pro trendový prohlížeč vždy proskytovat datové signály s mírným přebytkem vzorků.

Z předchozího Obr. 20 vyplývá, že takto provedená úprava pomocí půlení počtu vzorků ve výsledku znamená maximálně zdojnásobení počtu dat, které je potřeba uložit do paměti PC pro každý obsažený datový signál v aktuálně používaném datovém zdroji. Za tuto cenu jsou však pro trendový prohlížeč všechna data dostupná ihned. Jediná prodleva nastává v případě prvního zobrazení dat v datovém prohlížeči, kdy nastává stav přípravy dat. Následné procházení zobrazených křivek pak lze provádět plynule bez vlivu prodlev v části datového rozhraní, což bylo hlavním cílem tohoto řešení.

bool OnlineDataStart(), bool OnlineDataStop()
Tyto komunikační metody nejsou pro datový zdroj *CsvDataReader* využity.

5.5 Implementace knihovny pro přístup k datům k online datům

Tato podkapitola pojednává o realizaci vytvoření DLL knihovny pro online datový zdroj. V realizovaném datovém rozhraní se data z online datového zdroje získávají za pomocí využití dostupné komunikační metody pocházející z firmy Ingeteam a.s. [26]. (bližší informace uvedeny v 3.6 Komunikační prostředek firmy Ingeteam a.s.)

Rozhodnutí využití této komunikační metody bylo provedeno z několika důvodů.

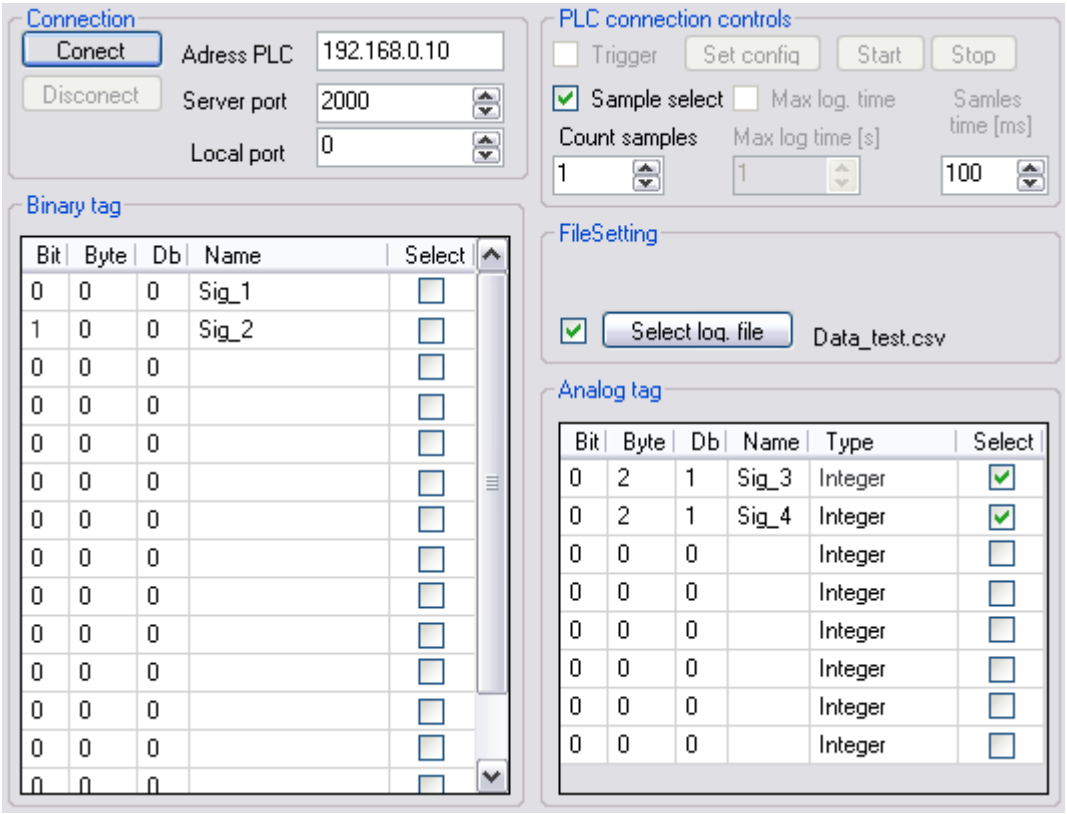
1. Komunikační metoda je ve vlastnictví firmy, není potřeba investovat do licencí komunikačních prostředků.
2. Metoda zaručuje zprostředkování z každého cyklu PLC. Proto je například možno získávat data z každého akčního zásahu regulátoru.

3. Původně byla komunikační metoda navržena jen pro ukládání dat do databáze. Jelikož byly pro tento program dostupné zdrojové kódy jazyka C#, tak jej bylo možno za pomocí některých úprav využít pro implementaci v realizovaném datovém rozhraní v části reprezentující online režim.

Uživatelské okno z konfigurace datového zdroje

Toto konfigurační rozhraní je vytvořeno z obdobných prvků, které jsou rozčleněny do jednotlivých skupin jako konfigurace pro CSV datový zdroj

Pro přehlednost byly tyto požadované konfigurace v GUI rozčleněny za pomocí komponenty „Group Box“ do následujících skupin, které pak obsahují tyto další ovládací komponenty „Button“, „Radio button“, „Label“, „Combo box“. Jelikož při vyčítání dat z PLC je nejdřív nutné provést konfiguraci datových signálů, bylo proto potřeba v GUI využít komponentu „SourceGrid“ použitou pro konfiguraci analogových a digitálních signálů.



Obr. 21 Konfigurační rozhraní pro přístup k Online z PLC datům

Použití „Users Controls“ pro konfiguraci online datového zdroje S7

V navrženém rozhraní je pro tuto konfiguraci nutné použít tabulku „Analog tag“ (možno konfigurovat 8 tagů) pro konfiguraci analogových hodnot typu „Integer, Double, Real“. Dále lze pak konfigurovat tabulku „Binary tag“ (možno konfigurovat 16 tagů) pro konfiguraci proměnných typu bool. U obou typů proměnných je při definici pro konfigurace nutno zadat tyto následující parametry, jedná se o bit, byte, číslo datového bloku, jméno tagu a zvolení, zda je tag aktivní pro následující měření.

Nahrání konfigurace do PLC se provádí tímto následujícím způsobem. V prvním kroku je nutno zadat IP adresu a „serverport“ PLC ze kterého budou data načítána. V případě komunikační metody „Netpro“ je nutno také vyplnit položku „Local Port“. Oba porty musí být odlišné. Po nastavení těchto parametrů se pro spojení z PLC může použít tlačítko „Connect“. Po stisku tohoto tlačítka dojde ke spojení s PLC. Stav spojení je signalizován barvou tlačítka „Conect“ (červenozelená/barva). Poté je možno v sekci „Connection controls“ provést nahrání připravené konfigurace signálů do PLC. Spuštění přenosu dat z PLC se provádí tlačítkem „Start“. Před stiskem tlačítka lze nastavit, zda bude požadováno měřit počet kolik vzorků, nebo se bude jednat o kontinuální měření. Po načtení zvoleného počtu vzorků se provede zastavení čtení dat. Poskytovaná data do trendového prohlížeče lze archivovat pomocí nastavení úložiště datového souboru „CSV“ ve skupině nastavení „File setting“. Před spuštěním měření je nutno nastavit položku „Sample time“. Tento parametr je důležitý správně pro generovanou časovou osu signálu.

Třída S7 Connection TCP

Tato třída obsahuje převzatou pro komunikaci za pomocí Ethernetového spojení. Třída „*S7CommunicationTcp“* obsahuje komunikační metody pocházející z firmy Ingeteam a.s. [26] Původní komunikační metoda byla použita pro ukládání dat do databáze, ale v tomto případě bylo potřebné dostupná data uchovávat a poskytovat trendovému prohlížeči. Proto bylo potřeba tento kód třídy pro načítání online dat upravit. Jednou z úprav bylo vytvoření kruhové paměťi neboli „Circular buffer“. Data pomocí této komunikační metody jsou z PLC poskytovány po dávkách. Například každých 10 sekund 100 vzorků požadovaného signálů. Každá tato datová dávka je uložena do pole kruhového pole dat. Jelikož se jedná o data, která jsou uložena v operační paměti PC, tak je toto pole (paměť vzorků) definována s maximálním možným omezením. Data jsou uložena v kruhovém poli a proto je možno trendovému prohlížeči poskytovat online data s možností o zpětný náhled na vzorky dat (v testovaném případě online režimu se jednalo o 50 000 vzorků zpět). Ostatní načtené vzorky lze pak prohlížet v režimu „Offline“ z dat, které byly uloženy při načítání dat do CSV souboru. Je proto vhodné, aby bylo v každém měření zvoleno úložiště CSV souboru (nastavení „File setting“).

Využití metod a událostí komunikačního předpisu ve třídě datového zdroje *S7OnlineDataReaderTcp*

Pro implementaci kódu do jednotlivých tříd platí stejná pravidla jako pro datový zdroj CSV. V jednotlivé metody pro online navrací obdobné datové typy jako v offline režimu, pouze využití některých metod je možné až po příchodu události data *DataUpdate.*

Implementace jednotlivých metod komunikačního rozhraní pro datový online zdroj je provedena následovně:

event EventHandler DataUpdate

Událost je vyvolána po každém příchodu dat z PLC. V případě, že je nastaven datový soubor pro ukládání dat, tak se provede zápis došlých dat do datového souboru. Trendový prohlížeč musí na aktuální událost zareagovat žádostí o nová data a vykreslením těchto nových dat.

event EventHandler DataRefresh
Událost je vyvolána po nahrání datové konfigurace do PLC. Informuje datový filtr a trendový prohlížeč o dostupnosti nových signálů.

bool CheckOnline()
V tomto případě jedná o online zdroj, tak metoda navrátí hodnotu „true“.

public string[] FindSignals(int aSigPos)
Metoda předá seznam signálu, který byl z parametrů GUI pro online režim zaslán do PLC.

public int CountValue (int aSigPos)
Metoda, která zjistí kolik je načtených vzorků dat z PLC

public double[] GetData (int aSigPos, int aSampleStep,int aStartIndex, int aEndIndex, string aTime)
Metoda, která poskytne dostupná načtená data z PLC uložená v kruhové paměti.

public string GetDescription()
Metoda navrátí popis „Online S7 Connection“.

bool OnlineDataStart(), bool OnlineDataStop()
Tyto metoy umožňují ovládat spouštění a zastavování načítání dat z aplikace trendového prohlížeče.

public bool Refresh(bool aCsvGuiRefresh), public bool Previous(),public bool Next()
public bool MakeDataSamples(string[] TimeType),
Tyto čtyři metody nejsou pro „online“ režim využity.

5.6 Implementace knihovny datového filtru

Knihovna datového filtru je poslední částí, která slouží v implementovaném datovém rozhraní k práci se zdrojovými daty. Pří načtení načítání především z offline zdrojů může dojít k nadbytku počtu vzorků, ale také k nadbytku datových signálů, které nejsou pro aktuální analýzu dat podstatné. Nadbytek počtu vzorků je řešen pomocí výběrů vzorkovacích kroků, start a stop indexů v rámci knihoven datových zdrojů.

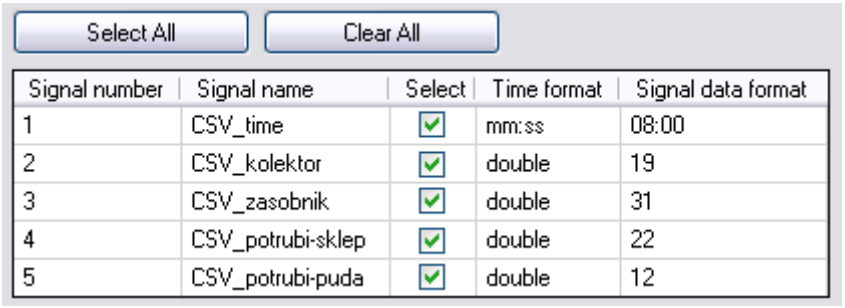
Z důvodu vyřešení problematiky filtrace nepotřebných signálů byla zavedena knihovna datového filtru, která pro práci s daty využívá také definovaný interface komunikačních metod *IDtataReader*.

Uživatelské okno z konfigurace datového filtru

Pro uživatelské okno bylo potřeba vytvořit přehledný seznam aktuálně načtených signálů. Byla použita komponenta „SourceGrid“ obdobným způsobem jako v případě GUI pro konfiguraci online režimu.

Dále pak pro výběr signálů byly použity dvě komponenty uživatelských tlačítek, které slouží k označení výběru pro všechny nebo žádné signály. Jedná se o tlačítka „Select All“a „Clear All“.

Navržená tabulka datového filtru obsahuje číslo signálu „Signal number“, jméno signálu „Name“. Dále komponentu „Check box“ sloužící pro výběr použití vybraného signálu v trendovém prohlížeči. Mezi další patří parametr „Time format“ sloužící pro uvedení v jakém datovém formátu je načten časový signál. Poslední sloupec tabulky „Signal data format“ slouží uživateli jako náhled na formát dat pro daný signál, aby byl informován o datovém formátu dostupného datového signálu.



Obr. 22 Ukázka datového „Users Controls“pro datový filtr s nastaveným formátem času

Použití „Users Controls“ pro konfiguraci online datového filtru

Použití uživatelského rozhraní pro konfiguraci datového filtru je jednoduché. Uživatel vybere potřebné datové signály a v případě potřeby zapíše parametr formátu časového údaje. Standardně je v sloupci „Time format“ vyplněno „double“. V případě, že se jedná o signál časového formátu, musí uživatel správně vyplnit řetězec formátu dle skutečného zobrazeného formátu pro načtený datový signál „Signal data format“.

Využití metod a událostí komunikačního předpisu pro třídu datového filtru

Třída datového filtru provádí pouze filtraci signálů dle uživatelského konfiguračního nastavení. Přes objekt datového filtru prochází všechna zdrojová data z datového zdroje do trendového prohlížeče. Většina metod pouze odkazuje na informace z datového zdroje bez žádných dalších úprav. U některých metod dochází k úpravám argumentů dle nastavení parametrů GUI datového filtru.

public double[] GetData (int aSigPos, int aSampleStep,int aStartIndex, int aEndIndex, string aTime);

Příklad implementované metody datového filtru, která požadavek modifikuje. V případě, že je v tabulce filtru vybrán například poslední signál, je změnit argument „aSigPoz“. Protože v trendovém prohlížeči bude tento veden jako první, ale v objektu zdrojových dat se musí přistoupit k poslednímu signálu.

Další modifikaci provádí třída filtru s argumentem „aTime“, který je v případě volání této metody z trendového prohlížeče vždy „double“. V případě, že tento parametr v GUI datového filtru pro daný signál změněn. Poté se v této metodě zavolá totožná metoda *GetData* s upravenými argumenty a referencí na objekt datového zdroje. Získaná data se pak poskytnou návratovou hodnotou trendovému prohlížeči.

Názorná ukázka kódu datového filtru pro metodu sloužící poskytnutí dat dle parametrizace filtru:

```
public double[] GetData(int aSigPos, int aSampleStep, int aStartIndex,
                        int aEndIndex , string aTime)
        {
            // Získání nových argumentů dle parametrizace
            // datového filtru.
            int lSigPos = mDataFilterConfig.SelectedInt[aSigPos];
            string lTime = mDataFilterConfig.TimeFormat[aSigPos];

            // Návratová hodnota se získá zvoláním metody na objekt
            // obsahující reference datového na datový zdroj.

            return mSourceDataReader.GetData(lSigPos, aSampleStep,
            aStartIndex, aEndIndex, aStartStopEnable, aSetSampleStep,
            lTime);
        }
```

6 *Praktické ověření datového rozhraní na vzorcích dat*

V této kapitole je pojednáno o testování datového rozhraní pro poskytování dat trendovému prohlížeči. Testy byly prováděny na současně vyvíjeném trendovém prohlížeči. V začátcích tvorby se jednalo pouze o základní zkušební testy koncepce řešení datového rozhraní na malém počtu datových vzorků. Pro testování offline datového zdroje byly použity různé CSV soubory závislé na typu testu. Jednotlivé testy se zaměřovaly především na testy rychlosti načítání dat, otestování a vlivu aliasingu a jeho odstranění. V poslední řadě se jednalo o testování předávání signálu dle požadovaného formátu času.

6.1 Testování správného zprostředkování offline dat pro trendový prohlížeč

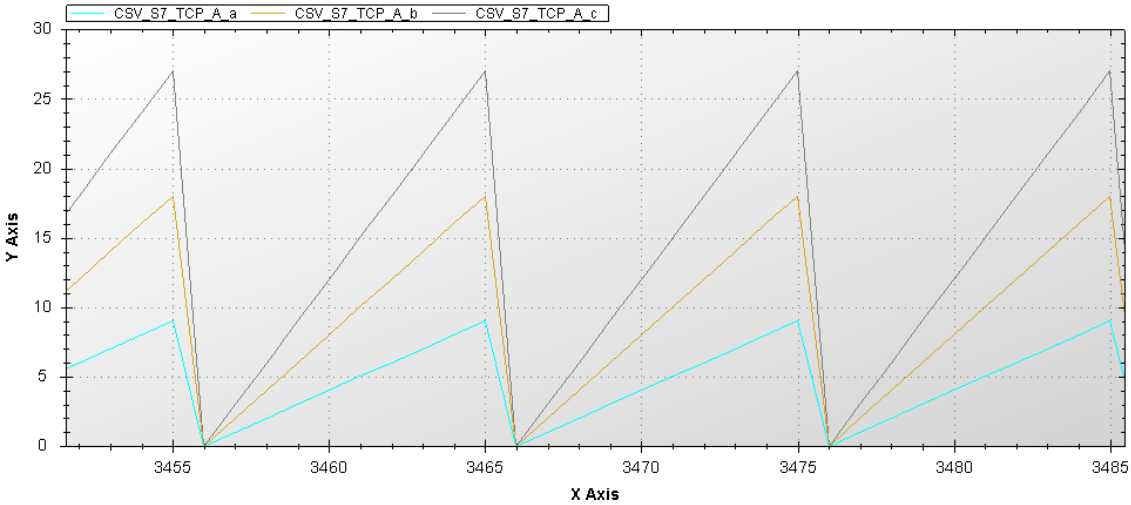
V této podkapitole jsou uvedeny výsledky zobrazení dat, u kterých při vykreslování velkého počtu vzorků hrozil vznik aliasingu. Test byl prováděn, jak na speciálně generovaných, tak skutečně naměřených datech.

Testování zkreslení signálu vlivem vysokého vzorkovacího kroku na periodickém signálu

Jak již bylo zmíněno při popisu realizace datového rozhraní, požadovaná perioda vzorkovacího kroku se určuje v trendovém prohlížeči na základě aktuálního nastaveného rozlišení obrazovky PC a počtu vzorků, které jsou obsaženy v požadovaném signálu. Pro níže uvedené testování bylo rozlišení obrazovky nastaveno na 1280 x 800 pixelů.

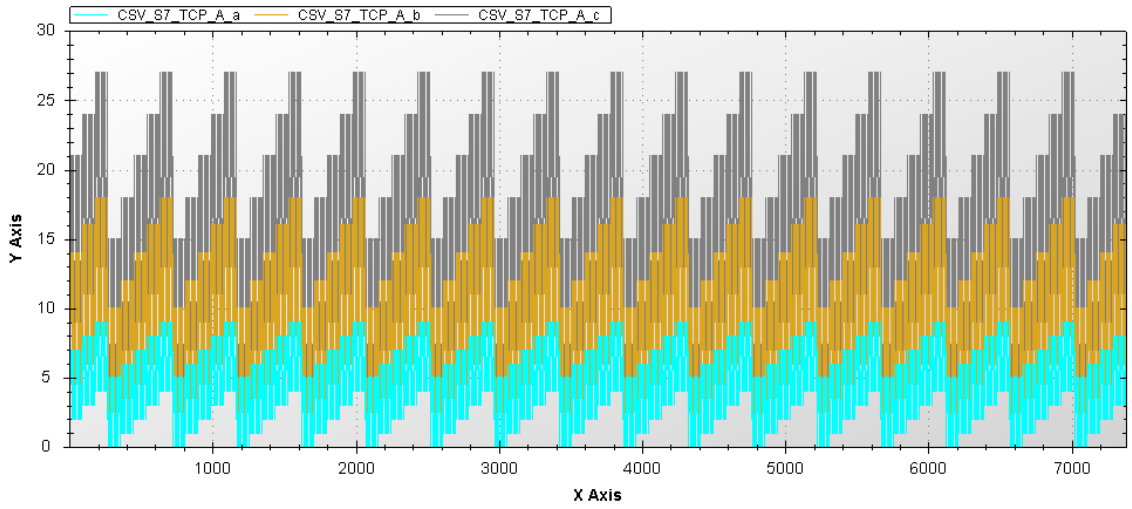
Na Obr. 23 je znázorněn správný tvar datových signálů. Jedná se o vykreslení části signálu, která obsahuje celkem 7380 vzorků. Data pro vykreslený signál byla poskytnuta trendovému prohlížeči za pomocí využití metody datového rozhraní „GetData“ s požadovaným argumentem vzorkovací periody o hodnotě 1.

Jelikož se jedná jen o vykreslení části signálu, byl tento požadavek o takto požadovaná data datovému rozhraní poskytnut pomocí metody „GetData“ obsahující argumenty metody „start a stop index“.



Obr. 23 Skutečný tvar analyzovaného signálu

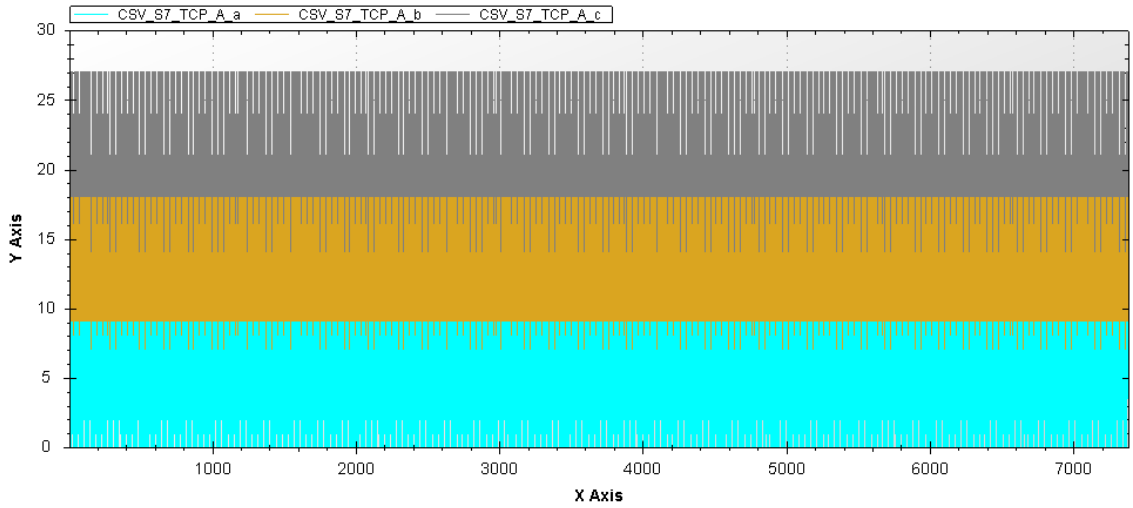
Na Obr. 24 je provedeno poskytnutí všech dat pro vykreslení celého náhledu signálu. Z obrázku je patrné, že původní tvar signálu je značně deformovaný a výsledná pravdivost informace o tvaru skutečného signálů je nulová (absolutně nevěrohodná data). Jedná se o stav, kdy je v GUI pro datový zdroj CSV nastaven parametr „Resolutiton setting“ na hodnotu „ACTUAL“. Výhoda této volby spočívá v rychlosti načtení dat. Tuto metodu načítání lze však využívat jen za situace, kdy by měl být předem znám předběžný průběh datových signálů. V datech se nesmí nalézat špičkové hodnoty a perioda zdrojového signálu musí být taková, aby vzhledem k zvolené periodě vzorkovacího kroku nezpůsobila aliasing, tak jak je uvedeno na Obr. 24.



Obr. 24 Deformovaný vykreslený signál

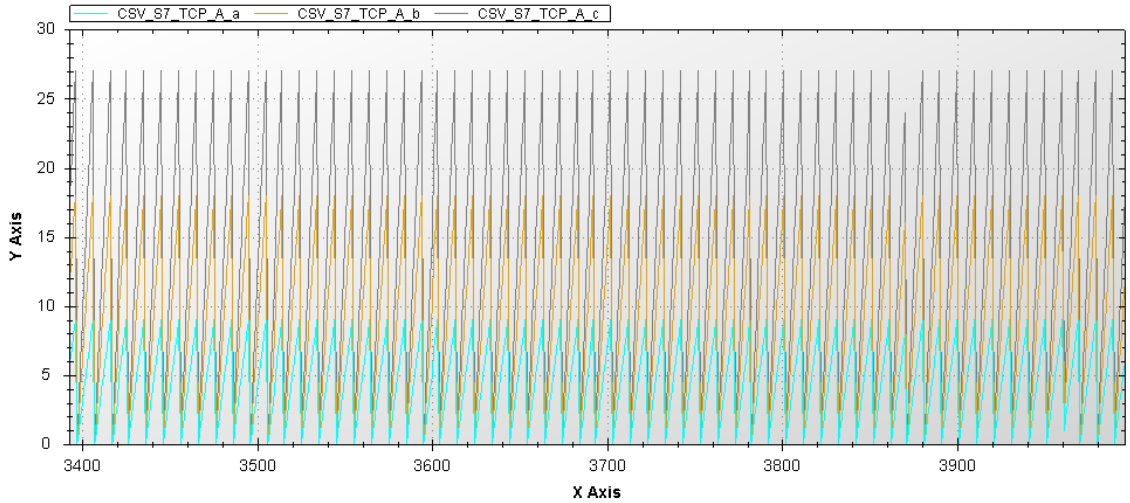
Na Obr. 25 je znázorněno správné vykreslení náhledu všech vzorků datového signálu. Pro tento způsob vykreslení bylo potřeba nastavit již výše zmiňovaný parametr „Resolutiton setting“ na hodnotu „MIN_MAX“. Tato volba způsobí, že je před prvním načtením signálu z datového rozhraní do trendového prohlížeče provedena analýza příprava a dat tak, aby nedošlo k vzniku aliasingu.

Nevýhodou je, že dojde ke zdržení před načítání dat z důvodu přípravy filtrace vzorků. Také nastane přibližně dvojnásobné využití operační paměti PC pro každý datový signál. Tohoto řešení však poskytuje výhodu v podobě stoprocentní správnosti vykreslení dat pro všechny očekávané volby period vzorkovacích kroků.



Obr. 25 Správně vykreslený signál – náhled na všechna data

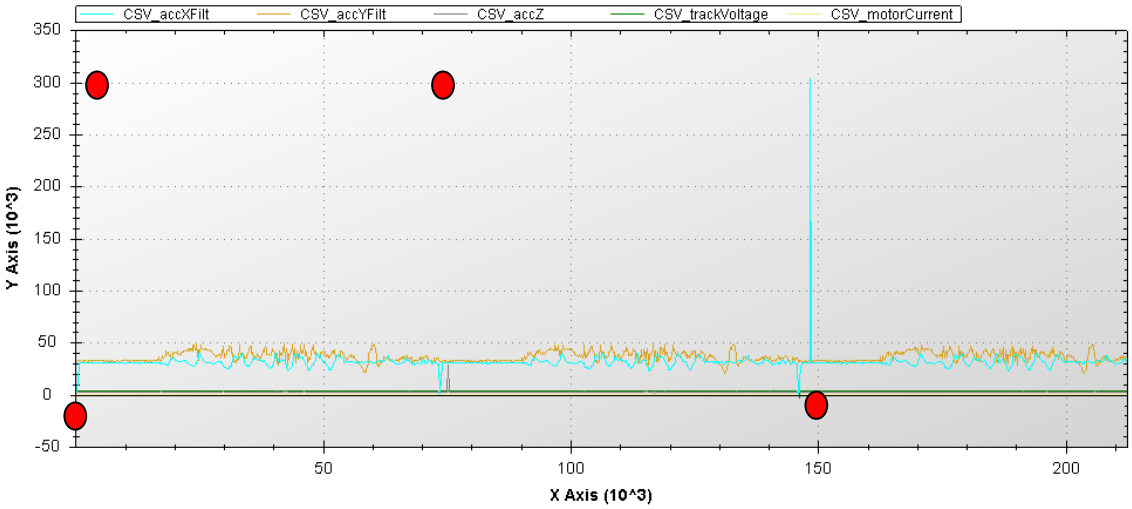
Z obrázku Obr. 25 vyplývá, že náhled na data je oproti náhledu na Obr. 24 je správný, ale příliš „hrubý“. Potřebuje - li uživatel bližší informace o tvaru signálu, tak se musí výběr oblasti vzorků dat (použití „zoomování“). Tímto „zoomovaním“ bylo nutno otestovat připravenost dat, pro jiné periodoy vzorkovacího kroku, viz. Obr. 26.



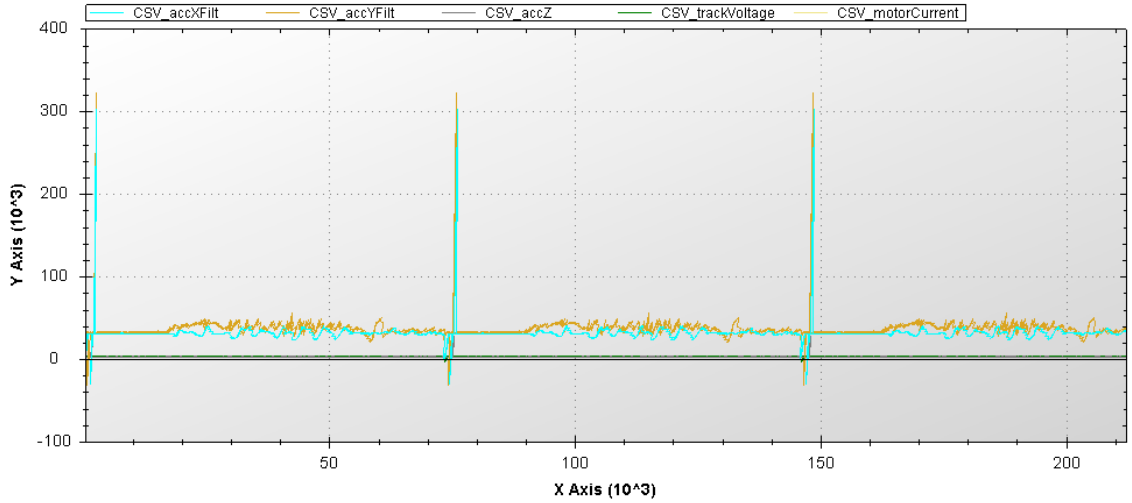
Obr. 26 Správně vykreslený signál - použití „zoomování“

Testování zobrazení špičkových hodnot signálu

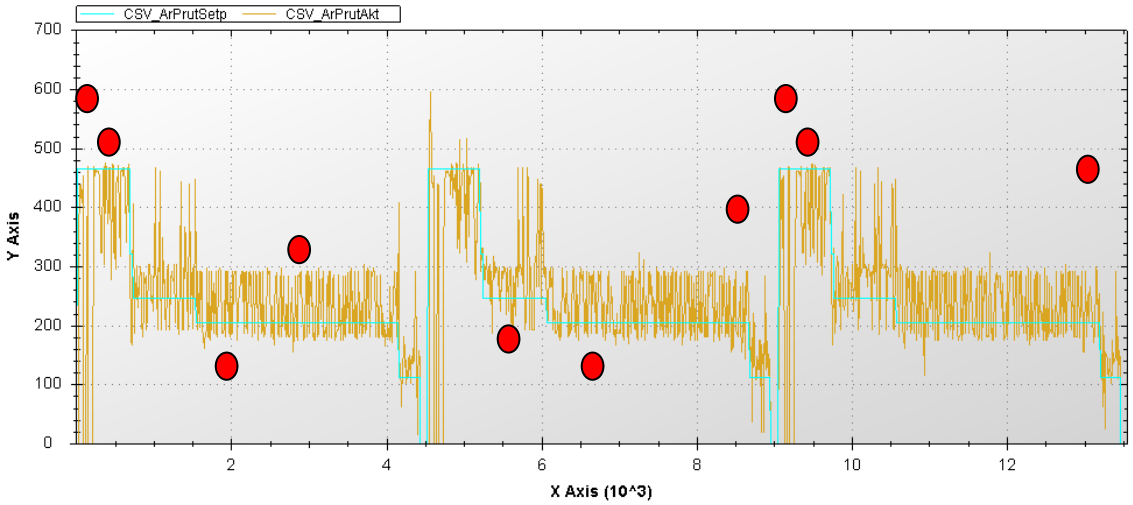
Metodika přípravy dat pro omezení výskytu aliasingu byla testována i na případech, ve kterých se v datovém signálu vyskytují špičkové hodnoty. Pro názornost je zobrazen příklad vykreslení dvou datových zdrojů obsahujících výskyt špičkových hodnot. Při porovnání nejsou rozdíly zobrazených dat při mezi nastavení „Resolutiton setting“ na hodnotu „ACTUAL“ a „MIN_MAX“ na první pohled patrné. Proto jsou v obrázku tyto chybějící vzorky dodatečně označeny (červené body).



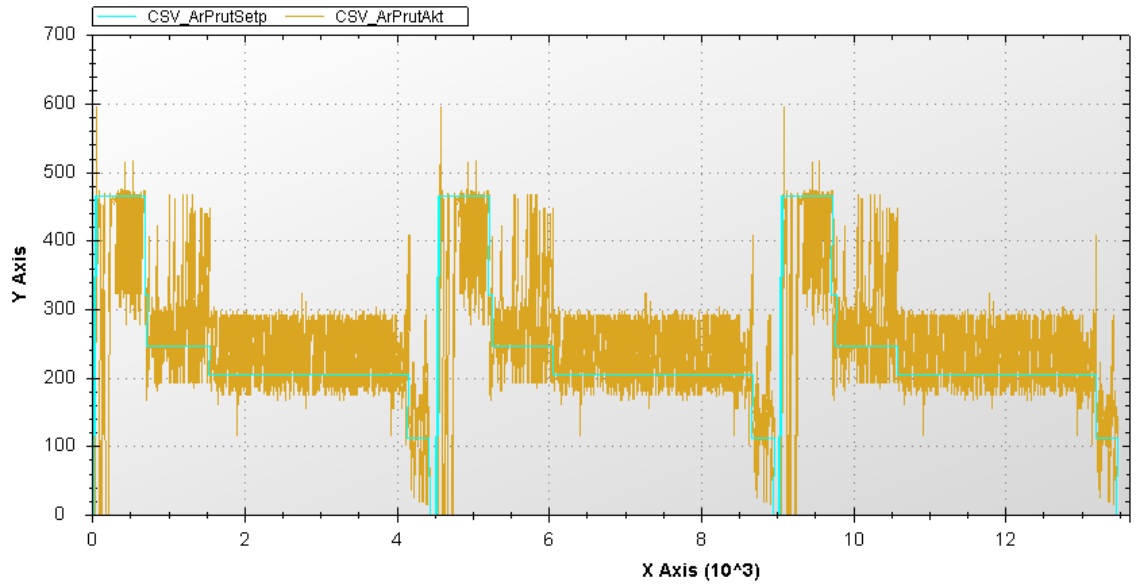
Obr. 27 Datové signály reprezentující náhodné špičkové hodnoty s nastavením „ACTUAL“- bez kontroly špičkových hodnot s označením chybějících vzorků



Obr. 28 Datové signály reprezentující náhodné špičkové hodnoty s nastavením „MIN_MAX“ - všechny vzorky správně vykresleny



Obr. 29 Datové signály reprezentující odezvu na jednotkový skok s nastavením „ACTUAL“- bez kontroly špičkových hodnot s označením chybějících vzorků



Obr. 30 Datové signály reprezentující odezvu na jednotkový skok s nastavením „MIN_MAX“- všechny vzorky správně vykresleny

Z uvedených obrázků je patrné, že navržená metoda pro omezení aliasigu funguje správně pro všechny typy datových signálů.

6.2 Testování poskytování online dat

Pro poskytování online dat bylo využito navrženého komunikačního způsobu pocházejícího z firmy Ingeteam a.s. Při testu bylo využito PLC Siemens typu CPU 317-2 PN/DP.

Bylo provedeno celkové otestování poskytování dat z online datového zdroje do trendového prohlížeče za pomocí realizovaného datového rozhraní včetně možností archivace dat do datového souboru CSV.

Formát ukládaných dat do CSV souboru

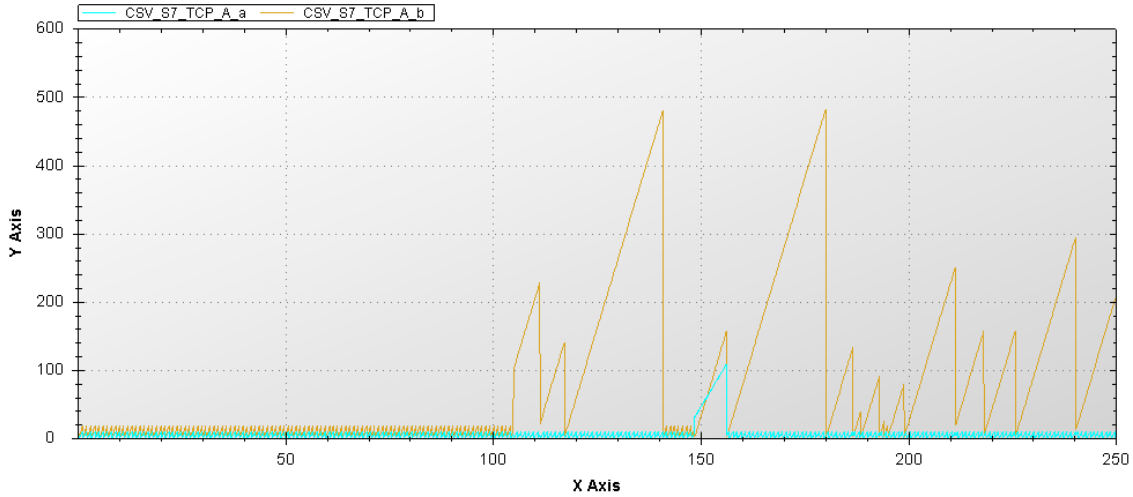
Aktuálně načítaná data lze ukládat do datového souboru CSV v následujícím tvaru:

Hlavička signálů	ID;	S7_TCP_A_a;	S7_TCP_A_b
Vlastní data	1;	1;	4;
	2;	2;	5;
	3;	2;	5;

Tyto data jsou zapisována po každém příchodu dávky dat z PLC.

Načítaná online data z PLC

Na následujícím obrázku je uveden příklad vykreslování online dat. Díky použité komunikační metodě bylo možno znázornit přesně data z každého cyklu PLC bloku OB35.



Obr. 31 Příklad vykreslování online dat

Data byla vykreslována po příchodu generované události z datového rozhraní, která poskytovala informaci o dostupnosti nových dat.

Při testování online dat byla odzkoušená možnost zpuštění dvou instancí trendového prohlížeče s realizovaným datovým rozhraním (na PC byla zpuštěna 2 x aplikace trendového prohlížeče). Datové rozhraní prvního trendového prohlížeče bylo nakonfigurováno pro načítání dat z online datového zdroj. Datové rozhraní pro druhý datový zdroj bylo nakonfigurováno pro offline zdroj.

Celý důvod tohoto testu byl následující:

Jak již bylo zmíněno, tak v online režimu je možno zobrazit pouze omezenou velikost načtených dat. Aby bylo možno při načítání online zobrazit data staršího vzorkovacího kroku, (data před 50 000 vzorkovacími kroky) tak bylo nutno tyto data archivovat do souboru CSV.

Zpuštěná aplikace prvního trendového prohlížeče s datovým rozhraním prováděla vykreslování online dat, která se ukládala do datového souboru.

Druhá aplikace byla nastavena pro získávání dat ze souboru, do kterého byla ukládána online data. Data z online režimu byla při testu aktualizována do datového souboru s periodou přibližně 10 s. Pro tato data je pak možno použít všechny možnosti nastavení datového zdroje „offline“ včetně algoritmu pro omezení aliasingu při náhledu na všechna data. Díky skutečnosti, že datové rozhraní obsahuje metodu pro aktualizaci dat *Refresh*, tak bylo v trendovém prohlížeči v offline režimu možno jedním jednoduše dle potřeby provádět aktualizaci dat, která jsou do datového souboru průběžně doplňována současně ze zpuštěné aplikace pro online datový zdroj.

Toto řešení je mírně nepohodlné, ale v rámci současného datového rozhraní a trendového prohlížeče splňovalo požadavky vykreslení požadovaných datových signálů.

7 Zhodnocení dosažených výsledků

Tato kapitola pojednává o zhodnocení celkového koncepčního návrhu a o zhodnocení výsledků testů datového rozhraní.

7.1 Zhodnocení koncepce realizace datového řešení

Úkolem této práce bylo navrhnout a realizovat návrh datového rozhraní pro datový prohlížeč reálných a historických trendů, za pomocí využití platformy .NET a programovacího jazyku C#.

Tento požadavek byl vyřešen návrhem datového rozhraní formou DLL knihovny a přesně definovaného komunikačního „interface“ vytvořeného pomocí komunikačních metod a událostí. Tento způsob realizace se později při vývoji a vytváření jednotlivých testovacích aplikací osvědčil jako dobrá volba, protože implementované rozhraní za pomocí knihovny bylo možno snadno navázat na odděleně řešenou realizaci trendového rozhraní.

Dalším rozhodnutím při realizaci bylo rozdělit samostatné datové rozhraní do jednotlivých DLL knihoven datových zdrojů a hlavní knihovny datového rozhraní, ve které byly použity knihovny datových zdrojů. Díky vytvoření této koncepce, umožňuje datové rozhraní možnost doplnění dalších datových zdrojů.

Pro uživatele se aplikace datového rozhraní jeví pouze jako jedno uživatelské okno, které obsahuje možnost konfigurace online a offline datového zdroje. Je však třeba uvést, že kód programu je rozdělen do pěti knihoven a celková implementace datového rozhraní v jazyku C# bez použité upravované knihovny pro získávání online dat obsahuje přibližně 2600 řádků kódu.

7.2 Zhodnocení implementace a přístupu k datovým zdrojům

Zhodnocení částí implementovaného offline datového zdroje

Pro reprezentaci přístupu k offline datovému zdroji bylo využito načítání dat z datového zdroje CSV. Z praktického testování vyplynulo, že poskytování dat dle požadavků start, stop indexů a voleb period vzorkovacích kroků je navrženo a implementováno správně.

Při postupném vývoji a průběžném testování datového rozhraní na jednotlivých verzích trendového prohlížeče bylo zjištěno, že je potřeba vytvořit, z důvodu optimalizace rychlosti dostupnosti dat, další metodu, která poslouží k předpřipravení dat, která omezí při vykreslení dat vzniku aliasingu. Při testech bylo zjištěno, že metoda funguje dle očekávání správně a tím bylo dosaženo eliminace všech chyb, které se týkaly vykreslování náhledu celého signálu. Proto je vhodné, aby uživatel, pokud možno, vždy využíval pro všechny neznámé signály, v případě načítání dat z offline datového zdroje CSV, nastavení parametru „Resolutiton setting“ nastavenou na hodnotu „MIN_MAX“. V případě specifických požadavků, může uživatel vybrat nastavení „MIN“ nebo „MAX“. Nastavení hodnoty „ACTUAL“ je vhodné použít jen v případě velkého množství signálů s mnoha vzorky, pro které nepotřebuje uživatel zobrazit celkový náhled na všechny vzorky, ale provádí pouze analýzu určité části vzorků, jejichž počet je menší než nastavení rozlišení obrazovky PC. Pro tento případ analýzy vzorků by příprava dat eliminující aliasing nebyla opodstatněná. Zbytečně by byla prodlužována doba pro přípravu dat v datovém rozhraní.

Zhodnocení částí implementovaného online datového zdroje

Pro komunikaci s tímto zařízením je použito přístupu ethernetového spojení typu TCP, klient (PC) - server (PLC). Požadovaná data v PLC jsou v každém cyklu programu ukládána do dvou vyhrazených oblastí paměti. Jakmile dojde k naplnění jedné oblasti paměti, tak jsou data vyslána po ethertetové síti do cílového PC. Během vysílání jsou data v PLC ukládána do druhé oblasti paměti, čímž je zabezpečeno, že nedojde ke ztrátě dat z žádného cyklu programu PLC. Díky využití tohoto komunikačního způsobu, bylo dosaženo možnosti získávat data také z přesností periody jednoho cyklu PLC, jako v případě komerčně dostupných analyzérů Auto SPy nebo PLC ANALYZER - pro 5.

7.3 Celkové zhodnocení použitelnosti realizovaného datového rozhraní

Cílem zadání bylo vytvořit návrh a realizaci konceptu datového rozhraní, které lze nadále rozšiřovat o další funkce implementované do jednotlivých knihoven. Právě koncept datového rozhraní a využití komunikačních metod byl otestován na realizaci přístupu k datům pro offline CSV zdroj a online TCP spojení s průmyslovým automatem PLC Simatic S7.

Pro firmu Ingeteam a.s je výhodné, že lze tuto koncepci datového rozhraní s trendovým prohlížečem využívat bez nutných registračních poplatků pro všechny firemní PC. Realizované datové rozhraní se však svými možnostmi v současné verzi nevyrovná komerčně dostupným produktům, rozhraní je ve firmě uvedeno režimu testování.

8 Závěr

Úkolem diplomové práce bylo vyřešit problematiku návrhu a implementace univerzálního datového rozhraní pro platformu .NET umožňující poskytování offline a online dat trendovému prohlížeči. Cíl tohoto zadání pochází z firmy Ingeteam a.s. Realizace řešení rozhraní byla prováděna zároveň s realizací návrhu samotného trendového prohlížeče. Před začátkem návrhu a realizace řešení bylo nutné získat specifikaci požadavků od firmy Ingeteam a.s.

Další informace o možných požadavcích pro datové rozhraní byly načerpány studiem komerčně dostupných řešení pro získávání a zobrazování dat. Zjištěné výsledy analýzy možností zobrazování dat formou trendů a možnosti přístupu k požadovaným datům jsou shrnuty v úvodních kapitolách této práce.

Po teoretických částech bylo potřeba shrnout všechny načerpané požadavky, aby bylo možno začít řešit návrh a implementaci datového rozhraní. Datové rozhraní muselo být navrženo tak, aby bylo jednoduše implementovatelné do zdrojového kódu trendového prohlížeče. Také bylo potřeba, aby navrhované řešení umožňovalo případnou rozšiřitelnost o načítání dat z dalších datových zdrojů za předpokladu využití jednotně standardizovaného přístupu ke všem datovým zdrojům.

Požadavek jednoduché implementace byl vyřešen rozhodnutím vytvořením datového rozhraní formou knihovny DLL, kterou pak trendový prohlížeč mohl snadno využít. Veškerá komunikace mezi konečnou aplikací trendového prohlížeče a datovým rozhraním byla navržena univerzálně tak, aby umožňovala možnost přístupu ke všem datovým zdrojům, za pomocí využití totožných přesně definovaných metod a událostí.

V naprogramovaném datovém rozhraní pomocí využití programovacího jazyka C# je realizace offline datového zdroje reprezentována přístupem k datovému souboru typu CSV. Metoda připojení k datovému zdroji online je reprezentována vytvořením připojení k PLC Siemens S7 za pomocí využití ethernetového připojení TCP connection.

Celé datové rozhraní bylo testováno v začátcích v rámci testovacích aplikací, později pak v rámci současně vyvíjeného trendového prohlížeče. Jako vzorky dat pro testování offline režimu byla použita vlastní data a reálné vzorky naměřených dat z technologického procesu. Na datech z datového zdroje typu CSV byly otestovány základní požadavky datového rozhraní, které se týkají offline datového zdroje. Dále bylo provedeno testování správného způsobu předávání velkého množství dat, kdy trendový prohlížeč požadoval data s různou periodou vzorkovacího kroku. Důležitou součástí bylo také otestování algoritmu pro zamezení vzniku aliasingu. Při testování datového zdroje také byly testovány ostatní celky datového rozhraní, jako datový filtr a možnost ukládání parametrů nastavení datového rozhraní.

Během testů datového rozhraní pro načítání z offline a online datových zdrojů do trendového prohlížeče byla ověřena funkčnost správně navržené struktury datového rozhraní. Po několika úpravách prováděných v rámci testování splnilo datové rozhraní základní požadavky, které byly specifikovány při návrhu řešení. Současný stav umožňuje pomocí implementovaného rozhraní načítat z CSV souboru a z PLC Siemens S7. Celá koncepce datového rozhraní je však navržena modulárně a proto lze do rozhraní jednoduše doplnit o rozšíření načítání dat z dalších jiných datových zdrojů. V této době je stávající verze datového rozhraní s trendovým prohlížečem firmou Ingeteam a.s. používána v testovacím režimu.

9 Použité zdroje

Literatura

[1] NAGEL, Christian. *C# 2005: programujeme profesionálně*. Vyd. 1. Překlad Jakub Mikulaštík, Petr Dokoupil. Brno: Computer Press, 2006, 1398 s. ISBN 80-251-1181-4.

[2] NAGEL, Christian. C# 2005: programujeme profesionálně. Vyd. 1. Překlad Jakub Mikulaštík, Petr Dokoupil. Brno: Computer Press, 2006, 1398 s. ISBN 987-80-251-2027-9.

Internetové zdroje

[3] ADTEC s.r.o – PLC Analysér. *ADTEC: PLC-ANALYZER pro 5.* [online]. 2011 [cit. 2012-12-02]. Dostupné z:<http://www.adtec.cz/cz/automatizace/plcanalyzer.php>

[4] AUTEM GmbH: PLC-ANALYZER – PLC logic analysis in no time. *AUTEM: PLC-ANALYZER pro 5.* [online]. 2013 [cit. 2013-03-10]. Dostupné z:<http://www.autem.de/index.php?site=en_sps>

[5] Analyzátor Simatic S5 AutoSPy : FOXON s.r.o.. *FOXON: Analyzátor Simatic S5 AutoSPy.* [online]. [cit. 2013-12-10]. Dostupné z: <http://www.foxon.cz/s5-software-analyzator-simatic-s5-autospy-c-142_183_177.htm>l

[6] BRÁZDA, Roman a BRAUN, Vlastimil. Automa:: Automatizace v současnosti nestačí. *Automa časopis pro automatizační techniku.* [online]. 19.7.2007 [cit. 2013-02-02]. Dostupné z: <http://www.odbornecasopisy.cz/index.php?id_document=31292>

[7] COMES Historian. *COMPAS AUTOMATIZACE.* [online]. [cit. 2012-12-02]. Dostupné z:<http://www.compas.cz/bcdaa0d3_0641_47f5_90fb_99d2e3b98be5.aspx>

[8] InTouch SCADA HMI. *Pantek.* [online]. [cit. 2012-12-05]. Dostupné z: <http://pantek.cz/produkty/intouch/>

[9] Procesní vizualizační systém SIMATIC WinCC- Industry Automation & Drive Technologies – Siemens. *SIEMENS.* [online]. © 2013 [cit. 2012-12-08]. Dostupné z: <http://stest1.etnetera.cz/ad/current/index.php?ctxnh=525faea5c2>

[10] Co je PROMOTIC. *PROMOTIC SCADA visualization software.* [online]. [cit. 2012-12-09]. Dostupné z:<http://www.promotic.eu/cz/pmdoc/WhatIsPromotic/WhatIsPromotic.htm>

[11] Referenční model ISO/OSI -sedm vrstev. *Referenční model ISO/OSI – sedm vrstev.* [online]. [cit. 2012-12-09]. Dostupné z: <http://pc-site.owebu.cz/?page=ISO-OSI-1>

[12] Počítačové sítě – Model ISO/OSI. *Počítačové sítě SITE.THE.CZ.* [online]. [cit. 2012-12-10]. Dostupné z: <http://site.the.cz/index.php?id=4>

[12] KRYŠŮFEK,Jan . . *Profibus.* [online]. [cit. 2013-04-22]. Dostupné z: <http://www1.fs.cvut.cz/cz/U12110/site/profibus/>

[13] Základní informace o průmyslové sběrnici PROFIBUS – část I. FOXON s.r.o.. *FOXON: Základní informace o průmyslové sběrnici PROFIBUS – část I.*.[online]. [cit. 2013-04-22]. Dostupné z: <http://www.foxon.cz/index.php?main_page=faq_info&fcPath=42&faqs_id=164>

[14] Základní informace o průmyslové sběrnici PROFIBUS – část II. FOXON s.r.o.. *FOXON: Základní informace o průmyslové sběrnici PROFIBUS – část II.*.[online]. [cit. 2013-04-22]. Dostupné z: <http://www.foxon.cz/index.php?main_page=faq_info&fcPath=42&faqs_id=164>

[15] Ethernet. *Ethernet.* [online]. [cit. 2013-05-01]. Dostupné z: <http://home.zcu.cz/~zanetah/>

[16] KOSEK, Rostislav . Automa :: Profinet – standard pro průmyslový Ethernet v automatizaci. *Automa časopis pro automatizační techniku: Profinet – standard pro průmyslový Ethernet v automatizaci.* [online].[cit. 2013-04-10]. Dostupné z: <http://www.odbornecasopisy.cz/index.php?id_document=30419>

[17] Automa :: Ethernet Powerlink – komunikace v reálném čase. *Automa časopis pro automatizační techniku: Ethernet Powerlink – komunikace v reálném čase.* [online]. [cit. 2013-04-12]. Dostupné z: <http://www.odbornecasopisy.cz/index.php?id_document=33500>

[18] Komunikace pomocí standardního rozhraní DDE. *PROMOTIC SCADA visualization software: Komunikace pomocí standardního rozhraní DDE.* [online]. [cit. 2013-04-18]. Dostupné z: <http://www.promotic.eu/cz/pmdoc/Subsystems/Comm/DDE/DDE.htm>

[18] Funkce DDE – Access – Office.com. *Office: Funkce DDE.* [online]. [cit. 2013-03-15]. Dostupné z: <http://office.microsoft.com/cs-cz/access-help/funkce-dde-HA001228819.aspx>

[19] Nastavení NetDDE. *PROMOTIC SCADA visualization software: Nastavení NetDDE.* [online]. [cit. 2013-03-16]. Dostupné z:http://<www.promotic.eu/cz/pmdoc/Subsystems/Comm/DDE/NetDDESet.htm>

[20] Co je OPC ? OPC Server ? OPC Klient ? FOXON s.r.o.. *FOXON.* [online]. [cit. 2012-12-10]. Dostupné z: http://<www.foxon.cz/index.php?main_page=faq_info&fcPath=29&faqs_id=91>

[21] OPC Server | Products | Kontron Czech. *MERZ: OPC Server.* [online]. [cit. 2013-04-20]. Dostupné z: <http://shop.kontron-czech.com/default.asp?CatID=394>

[22] Platforma .NET Framework. *Windows Server: Platforma .NET Framework.* [online]. [cit. 2013-04-20]. Dostupné z: <http://technet.microsoft.com/cs-cz/library/cc738855(v=ws.10).aspx>

[23] Robust industrial strength data connectivity with MatrikonOPC Telemetry PLC Driver Suite. *MatrikonOPC.* [online]. 2013 [cit. 2013-04-05]. Dostupné z:http:<//www.matrikonopc.com/opc-drivers/plcs.aspx>

[24] Graybox – OPC Developer Tools (OPC Toolkit, OPC Simulator, OPC Development Tools, Documentation). *GRAYBOX: sowtware.* [online]. © 2002-2013 [cit. 2013-04-05]. Dostupné z: <http://gray-box.net/opc_tools.php?lang=en>

[25] QuickOPC – OPC Labs. *OPC Labs: Professional OPC Development Tools and Services.* [online]. © 2007–2013 [cit. 2013-04-06]. Dostupné z:http:<//www.opclabs.com/products/quickopc>

Firemní dokumentace

[26] DEDERLE, Pavel. *Manual – measure dream.* : Ingeteam a.s., 2008.

10 Seznam příloh

A. Rozšířený anglický abstrakt

The diploma thesis is focused on design and implementation of universal data interface for actual and historical trending in .NET.

Clear data displaying in form of data trends is one of the important requirements in industrial automation. The name – data displaying means the storing of values with timestamp, which is then used for graphical displaying of process values from PLC in a screen of PC. The values can represent for example the states of gas pressure in pipeline, the temperature in ovens etc. The required values are then displayed in "x ,y" graphs, in which the values of axis "x" are data, which in the most cases have the information of time in milliseconds, seconds, hours, days etc. The values of axis "y" are the values of signal. In industrial automation the graphs can be according to the obtained data divided into two basic groups. Online trends are the first group, the word online means that there are displayed data of actually running process. Offline trends are the second group; and they are used for displaying of historical data from control system, which are read from offline store – CSV file of SQL database etc.

The topic of the diploma thesis came from the Ingeteam a.s. company, which is focused on industrial automation. During commissioning of PLC systems, the instruments, which are used to monitoring process values in controlled technological processes, are important. Nowadays, professional data interfaces are used in industrial automation e.g. Iba analyzer, SPS Analyzer, PLC AutoSpy etc. These professional instruments have a wide range of options for an optimal displaying of data from industrial viewers. The big costs of licenses of data viewers are usually the disadvantage of these instruments. The big costs of licenses could limit the some buyers. This fact is one of the idea for creation our own data viewer according to the specified requirements. The design and creation of data viewer includes the wide range issues e.g. proper data retrieval from various data source, choice of required signals, data formatting for displaying in graphs, work with data, own manipulation with graphical viewer etc. The viewer development was divided due to sophistication into three diploma theses. The submitted diploma thesis deals with creating of data interface. The task of the interface is to provide data from data sources to trends' viewer. The next two diploma theses deal with displaying and data processing and for proper work they need the interface, which is solved in this diploma thesis.

During the solving issue of design and realization of data interface for trends' viewer it was needed to proceed in several steps. In the first step before design and realization, it was necessary to do analysis existing commercial products for data displaying from control systems in graphs. In industrial automation the displaying and data processing from control systems could be performed using many software products. In this diploma thesis the software products are divided into two basic groups according to the options of displaying graphic courses. The graphic displaying of data using programs of type "Data analyzer" forms the first group. The option of data displaying using graphic trends components contained in visualization software of type HMI is the second group. The dividing into the two groups is done due to various options and requirements of data displaying using trends. Data between PLC systems and trend software, which runs in PC, is mediated by using physical communication layer on ethernet base or communications on base of Profibus net.

The programs of “Data analyzer” type in most cases serves to programmers and technicians for commissioning, installation and testing of programs for industrial applications, which are controlled by industrial automats. It is known, that industrial applications, which are controlled by PLC systems, are usually time-critical processes. For this reason the high demands in speed of controlling are placed on industrial automats, e.g. the time-critical sections as feedback control or response of PLC to the emergency situations. From commercial programs of “Data analyzer” type e.g. PLC-ANALYZER pro 5 and AutoSpy it was found, that for data analyzers some requirements are important as an amount of transported data, the response speed and proper displaying and transfer of all required values from every PLC program period.

The second described group are the possibilities of displaying of data process using of systems HMI, which serves especially for controlling of technologies by operator. HMI systems obtain data usually from PLC using instrument as OPC client/server, DDE interface etc. The usage of these instruments supports the communication with various PLC systems and many types of visualisation systems. The disadvantage is found in some cases, which bounds communication response to usually minimal value 100 ms. So it means, that this systems are not able to display data with proper sampling frequency, e.g. according to the cycle time. Therefore, it is necessary during the data displaying by trends choose proper visualisation system for displaying of trends according to the requirements of signal for the application.

After the theoretical parts, which describe current issue and options of data viewers, the next parts are devoted to the design and realization of data interface. Due to the task of diploma thesis the data interface in .NET was created. To create programs using this technology features the wide range of object-oriented programming e.g. Visual basic .NET, C# etc. The data interface for trend viewer was designed using object-oriented language C# in program environment Microsoft Visual studio 2008. Before the implementation of program for data interface for trend viewer using language C#, it was needed to perform analysis of requirements and design of solution.

The following requirements were specified for data interface, which comes from studied issue and commercial data viewers:

The data interface has to be easily implemented to source code of data viewer.

Allow the possibility of widening of data retrieval from more data sources assuming usage of uniformly standardized access to data.

The application of data interface has to contain easy user configuration of parameters. And its repeated usage.

The data interface has to allow during the retrieval of big amount of data optimization of sampling step according to the possibility of rise of aliasing (a phenomenon which is observed when the low sampling frequency is selected for displaying of data signals).

From the requirements described above, it was possible to specify the design of realization of data interface using .NET and programming language C#. The creation of data interface using Dynamic-link library was the basic decision and it guarantees easy implementation to source code of trend viewer. All communication between final applications of trend viewer is designed universally and in way, which allows access to many data sources using exactly defined methods and events.

Designed program of data interface was according to the required functionality divided into several libraries. The main library – “DataProvider” includes interface (prescription of methods

and events which serves for assessment which communication instrument between data interface and application of trend viewer should be used). The prescription then has to use all other libraries with classes of data interface working with source data. In proposed data interface there are class as e.g. “CsvDaraProvider” or “OnlineProviderS7Tcp” served for obtaining data from offline or online data sources and “DataFilter”. For proposed data interface it was needed to solve issue of configuration of data interface. From this reason the library contains separate class for overall parameterization of interface and parameterization of individual sources. Apart from these basic classes the library contains more classes for guaranteeing the running of interface or storage of parameters of data source. For data retrieval to trend viewer classes in following order are used - the class of access to data sources and the class of filter of signal.

In programmed class of data interface the offline source is represented by the access to data file of CSV type. The method of online connection to data source is represented by the connection to PLC Siemens S7. For communication with this device the access by ethernet interface of TCP connection type, client (pc) – server (pc) is used. The principle of communication is following: The connection PC-PLC is established, and then the configuration data and information about required transmitted signal are sent. The selected communication method supports acquisition of data from every PLC program period. Required data from every PLC program period are stored in PLC in two reserved memory area. When one area of memory is full, data are sent using ethernet to target PC with running data interface and trend viewer. During the sending, data are stored to the second memory area in PLC, which guarantees, that no loss of data can occur in any PLC program period.

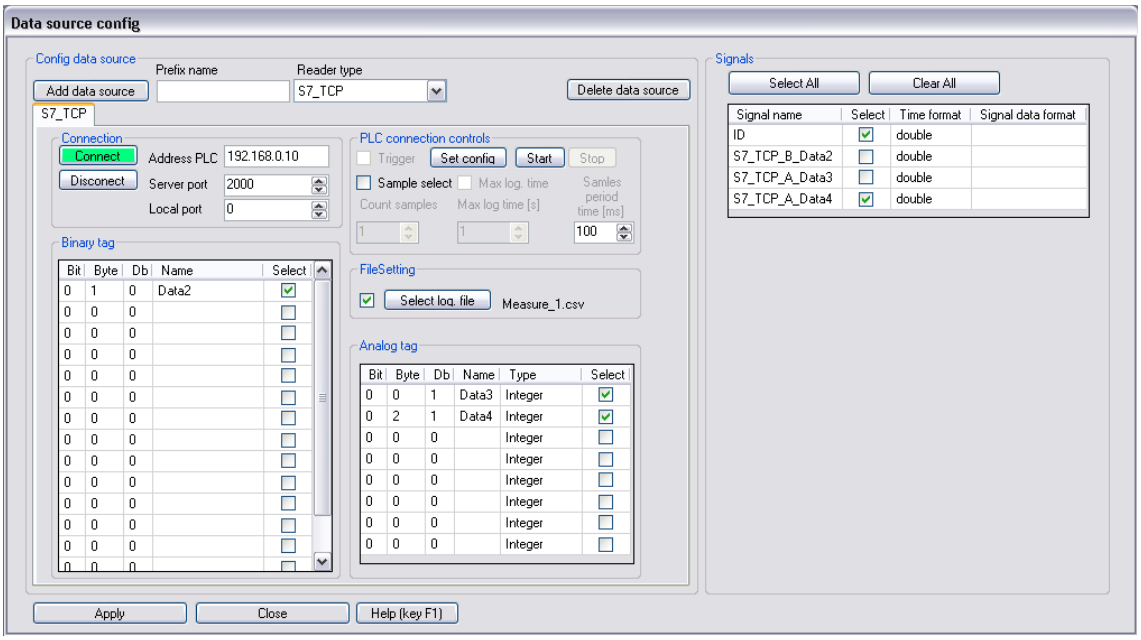
During the data retrieval in offline or online mode, they could be modified by program, which is contained in next classes. The class of data filter allows elimination of some signals, which are a part of data source, but it is not necessary to use them in data viewer.

The data interface was tested in test application and with parallel developed trend viewer. As a data samples for offline mode the data were generated and real samples of measured data from technological process. On data from data source of CSV type the basic requirements for data interface were tested. The ways of communication between data interface and trend viewer, proper transmission of large amount of data, where he setting of sampling step according to the prevention of aliasing, were tested. During testing of this data type the other parts of data interface were tested as data filter and storage of parameters of data interface.

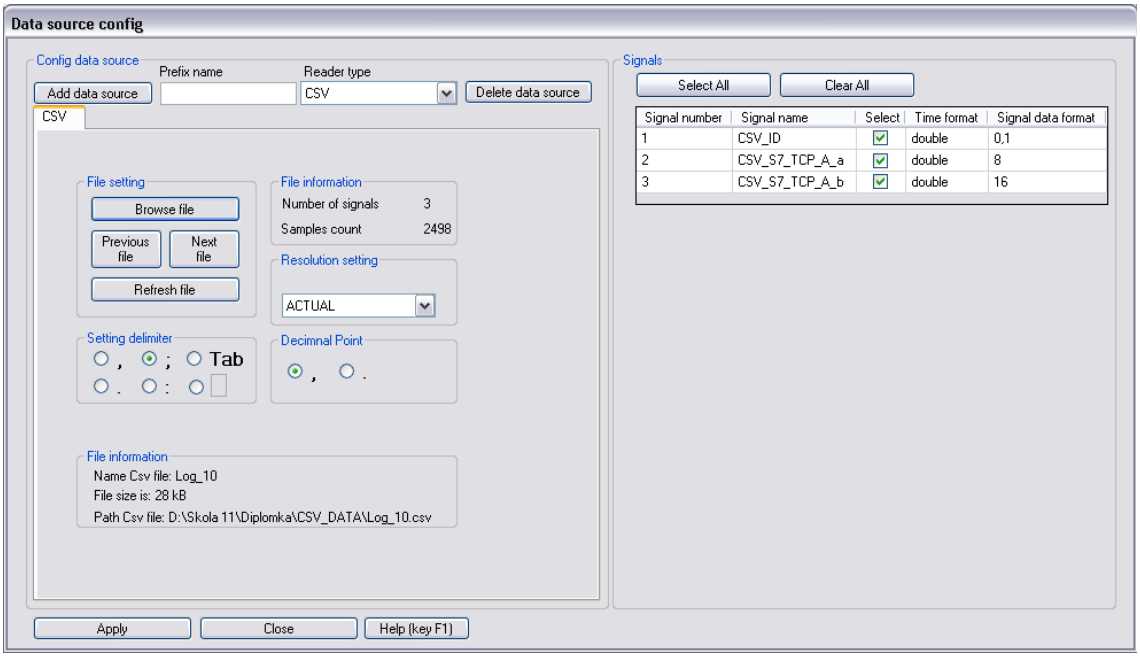
The testing of online data source was performed using Ethernet connection to PLC Simatic S7, which generated specified run of data signals. During the test it was verified, if the data interface is able to store and properly provide data from PLC to trend viewer.

During the tests of data interface for retrieval from offline and online data sources to trend viewer the functionality of designed structure of data interface was verified. The developed interface using programming language C#, after some adjustments, fulfilled the requirements, which were specified during the design of the solution and it allows retrieval data from CSV file and from PLC Siemens 7 too. The whole conception of data interface is designed in modular way and it allows simple widening of data retrieval from different data sources. Nowadays, the current version of data interface with trend viewer is in testing mode in Ingeteam a.s. company.

B. Konfigurační okno parametrizace datového rozhranní

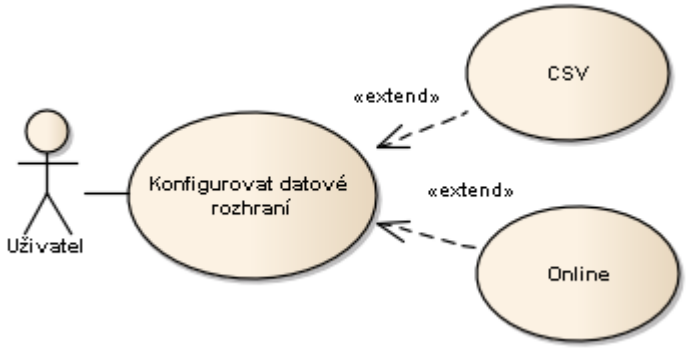


Obr. B.I Konfigurační okno parametrizace datového rozhraní - znázorněný příklad konfigurace pro online datový zdroj



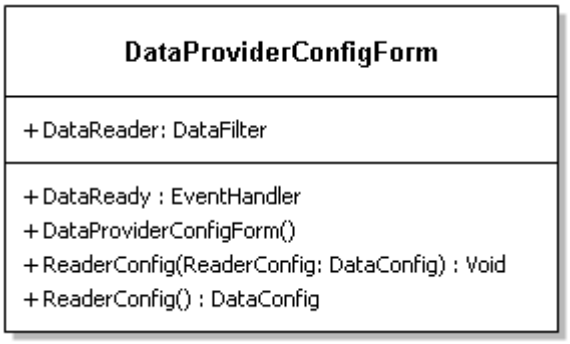
Obr. B.II Konfigurační okno parametrizace datového rozhraní - příklad konfigurace pro offline datový zdroj

C. Diagram případu užití datového rozhranní rozhraní

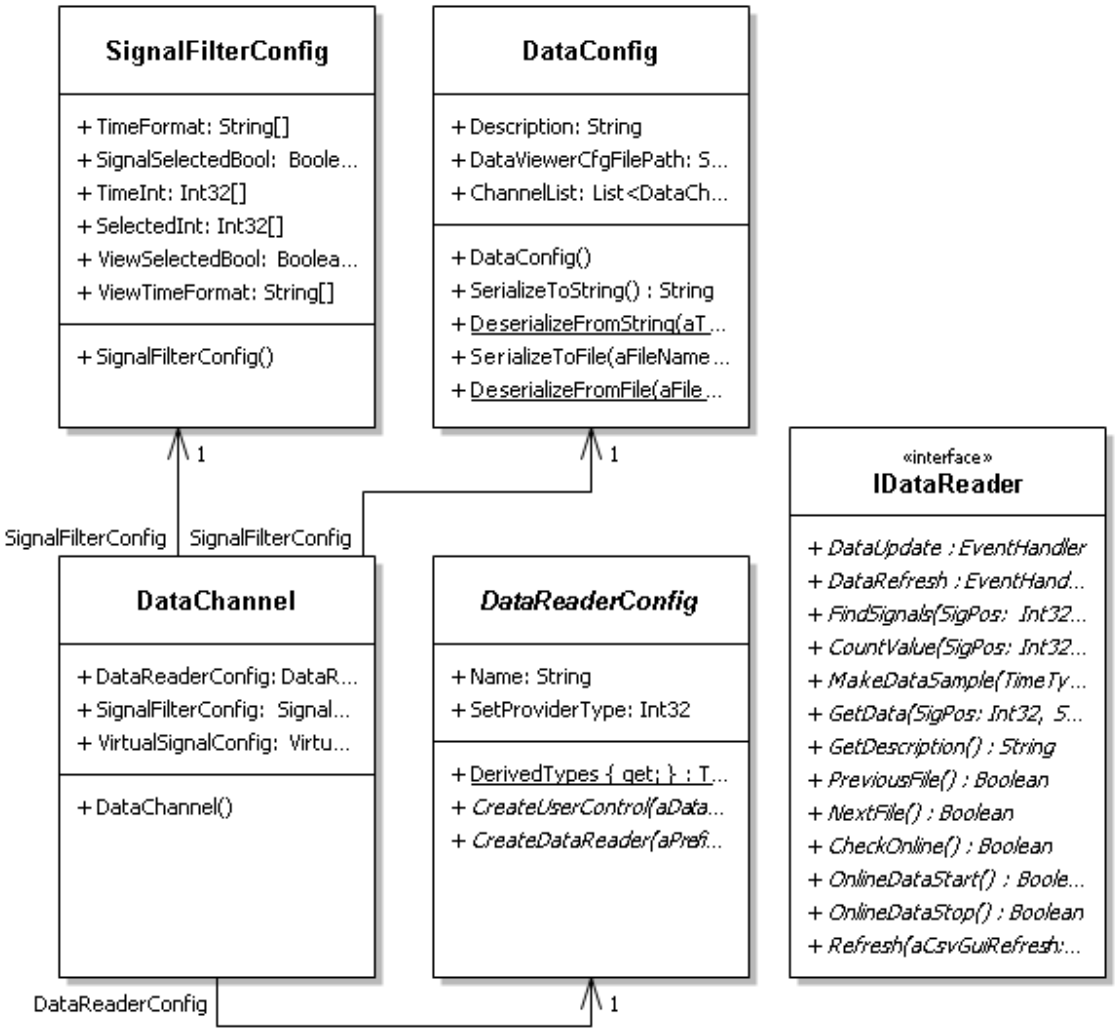


Obr. C.I Diagram případu užití

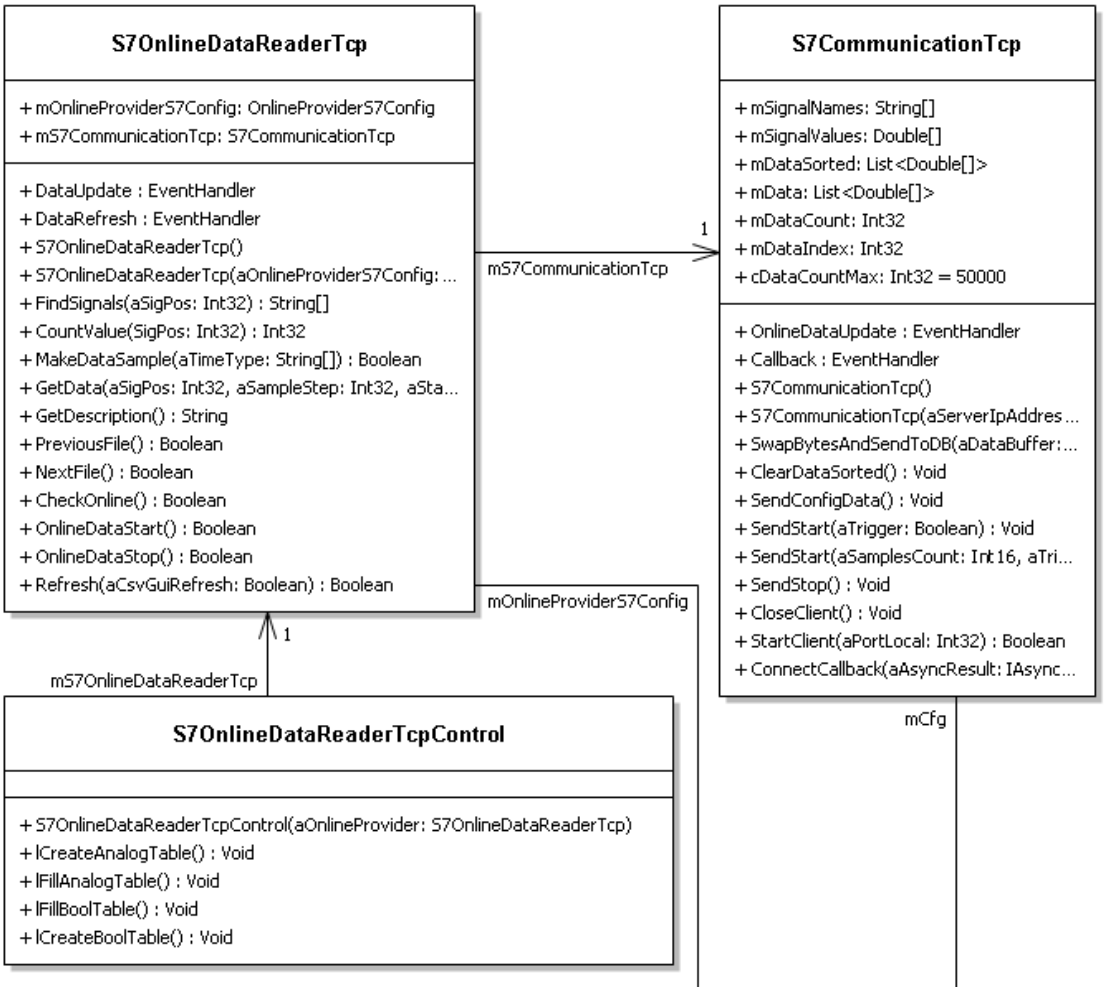
D. Diagram tříd



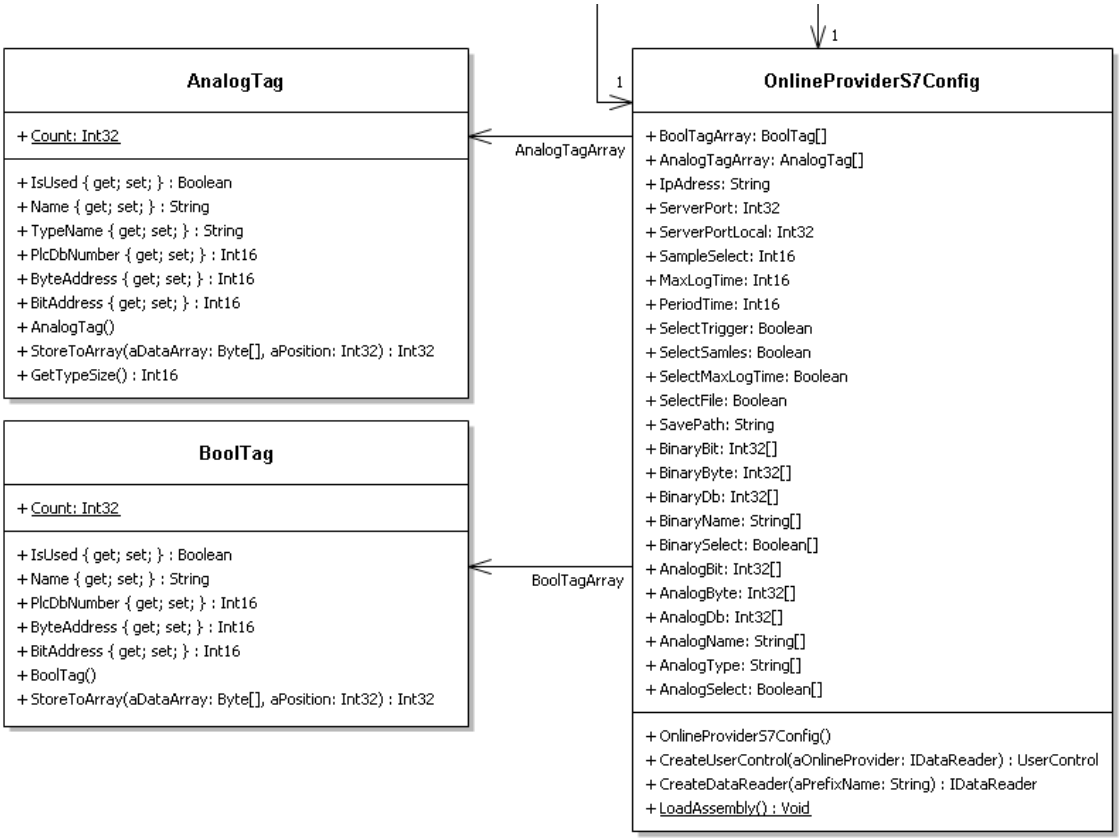
Obr. D.I Diagram třídy pro knihovnu DataProviderConfig



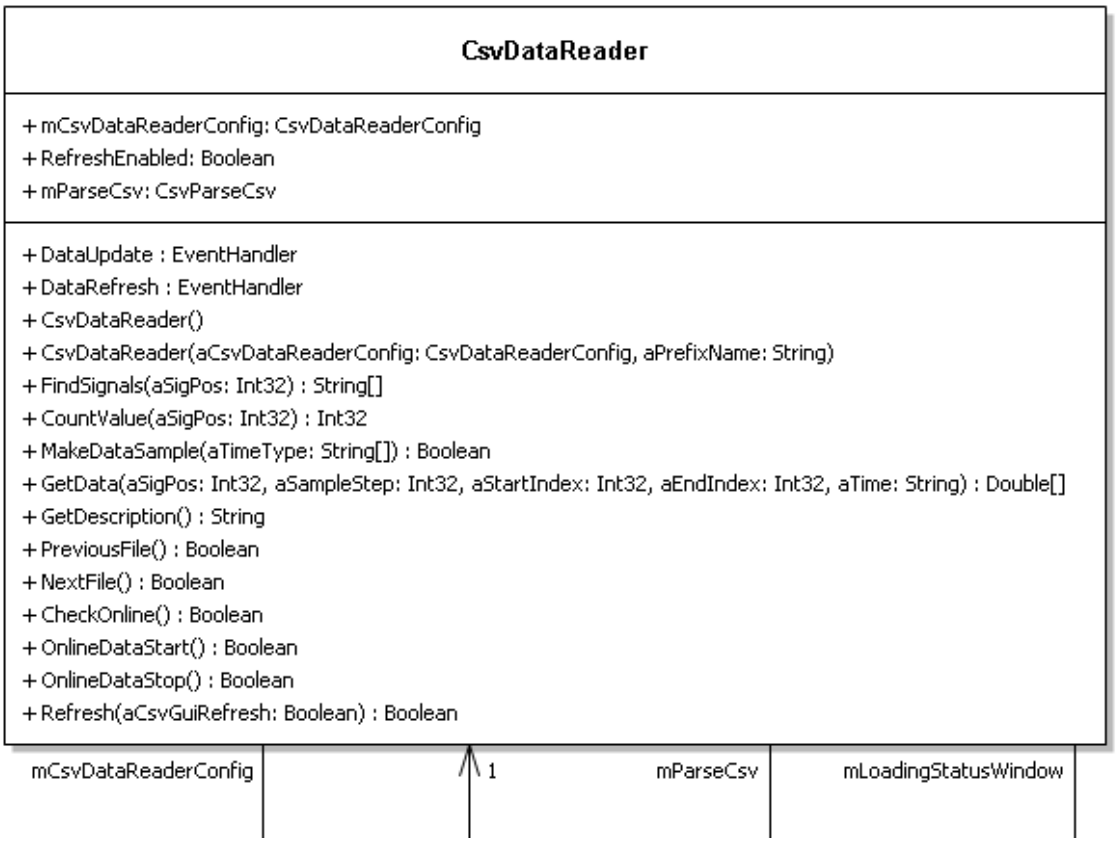
Obr. D.II Diagram tříd knihovny DataProvider



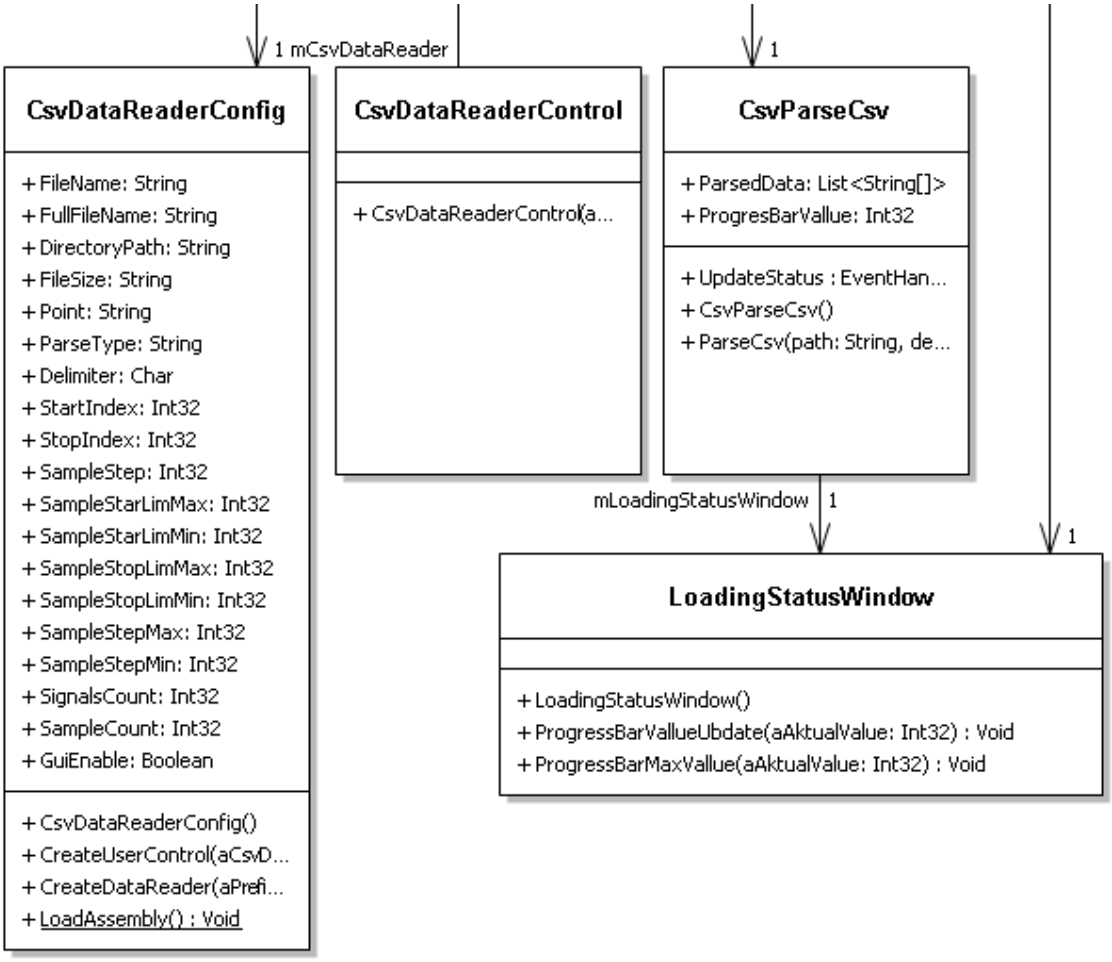
Obr. D.III Diagram tříd knihovny OnlineProviderS7Tcp(1/2)



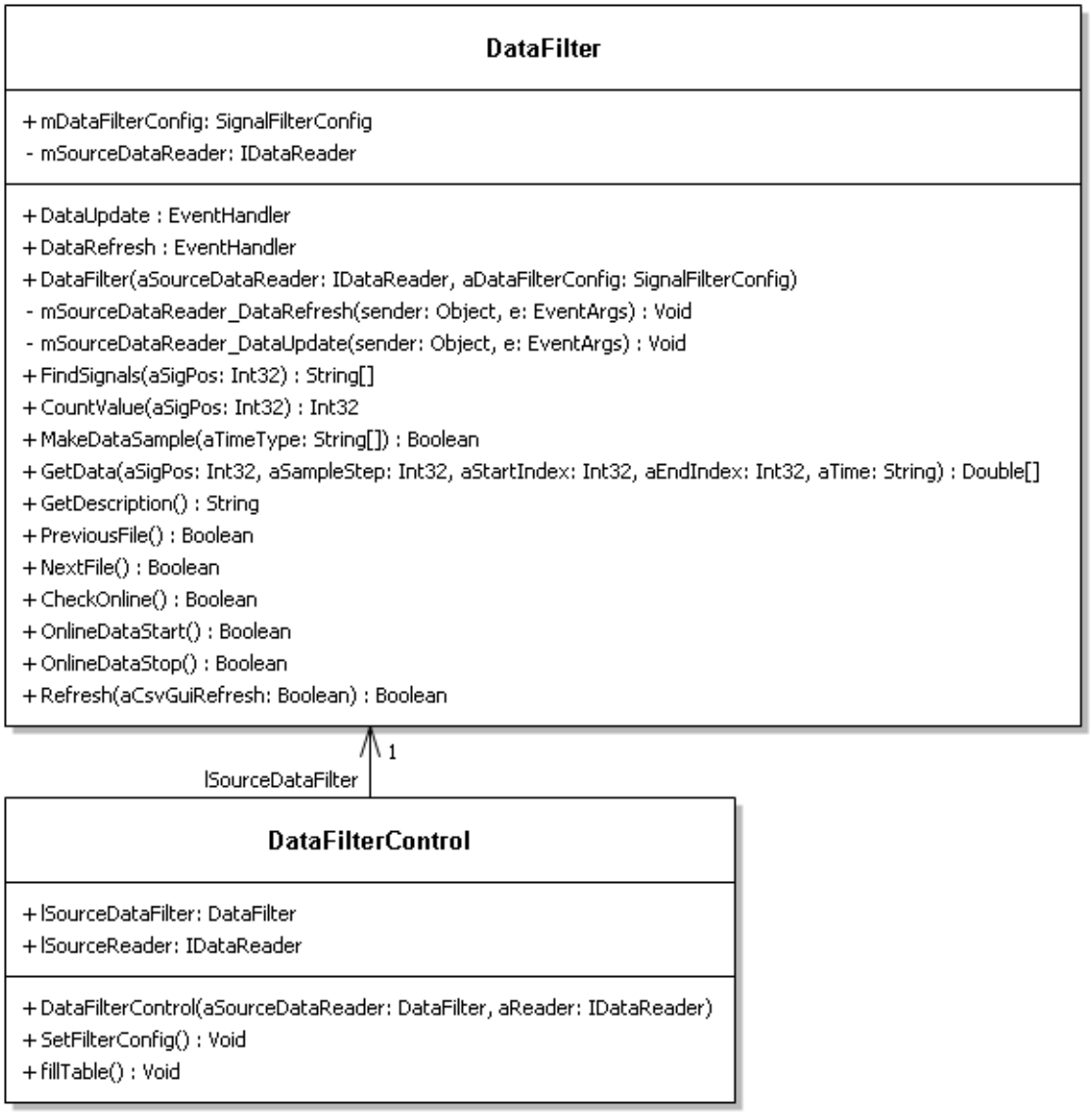
Obr. D.IV Diagram tříd knihovny OnlineProviderS7Tcp(2/2)



Obr. D.V Diagram tříd knihovny CsvDataProvider (1/2)



Obr. D.VI Diagram tříd knihovny CsvDataProvider (2/2)



Obr. D.VII Diagram tříd knihovny DataFilter

E. Obsah přiloženého DVD

Obsah přiloženého DVD je rozdělen do jednotlivých adresářů:

Diplomová práce	-	Obsahuje PDF s obsahem práce.
Realizované rozhraní	-	Obsahuje ukázkovou spustitelnou trendového prohlížeče, který využívá realizované datové rozhraní.
Zdrojová data	-	Obsahuje ukázková zdrojová data pro ukázkovou aplikaci.
Třídy DLL	-	Obsahuje jednotlivé znázornění tříd ve formátu PDF
Okna aplikace	-	Obsahuje obrázky konfiguračních oken datového rozhraní